ABSTRACT

Title of dissertation: EXPRESSIVENESS OF DEFINITIONS AND EFFICIENCY OF CONSTRUCTIONS IN COMPUTATIONAL CRYPTOGRAPHY

David Omer Horvitz
Doctor of Philosophy, 2007

Dissertation directed by: Professor Virgil Gligor
Department of Electrical and Computer Eng.
Professor Jonathan Katz
Department of Computer Science

The computational treatment of cryptography, and indeed any scientific treatment of a problem, is marked by its definitional side and by it constructive side. Results in this thesis better our understanding of both: on one side, they characterize the extent to which computational definitions capture the security of the basic task of symmetric encryption; on the other, they provide explicit bounds on the efficiency of commitment and secure two-party computation constructions. Specifically:

- We relate the *formal* and *computational* treatments of symmetric encryption, obtaining a *precise* characterization of computational schemes whose computational semantics imply their formal semantics. We prove that this characterization is strictly weaker than previously-identified notions, and show how it may be realized in a simpler, more efficient manner.

- We provide lower-bounds on the number of times a one-way permutation needs to be invoked (as a "black-box") in order to construct statistically-binding commitments. Our bounds are tight for the case of *perfectly*-binding schemes.

- We show that the secure computation of *any* two-party functionality can be performed in an optimal *two* rounds of communication even in a setting that accounts for concurrent execution with other protocols (i.e., the Universal Composability framework). Here, we rely on the assumption that parties have access to a common reference string; some sort of setup is known to be necessary.

EXPRESSIVENESS OF DEFINITIONS AND EFFICIENCY OF CONSTRUCTIONS IN COMPUTATIONAL CRYPTOGRAPHY

by

David Omer Horvitz

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2007

Dedication

To my father, Uri, my mother, Zila, and my sister, Naama.

Acknowledgements

My first thanks goes to Virgil Gligor. Whenever I think of where to start thanking Virgil, I go to the morning I gave my first conference talk, at CRYPTO. We had gone over my just-completed slides (this would become a signature move of mine), and I told Virgil I was nervous. After assuring me that it was a talk like any other and giving me advice about proper pace, Virgil said he would sit in the front row of the conference hall and suggested I looked towards him when I spoke. And so it was. Virgil is a first-class researcher, with a keen taste for quality and a remarkable ability to identify the next interesting area before everyone else. He has deftly guided my education and directed my development as a researcher. But above all, Virgil has been a caring friend that made me feel safe and accepted. He believed in me, has done above and beyond to support and promote my career, encouraged my somewhat naive and idealized ways, and shared his thoughts and experiences with me in and outside school. I am extremely fortunate to have walked this journey with him.

My second thanks is to Jonathan Katz. Jonathan joined the department when I was half-way into my program and had a great effect on my development. Jonathan gave order to the mass of definitions and results of modern cryptography through clear and dedicated teaching. He helped guide me towards theoretical problems, and taught me how to ask good questions about them (I may have once made one when I talked with Yevgeniy Dodis, as he asked if I worked with Jonathan as a response). Through invaluable interactions and his own work, Jonathan taught me how such problems should be tackled. Jonathan is a brilliant researcher, and it has been a privilege to work with him.

I thank my committee members for their steadfast support. I thank Aravind Srinivasan for his care for my progress and for suggesting specific tools for problems I was looking at. I thank Udaya Shankar for his guidance at the outset of my graduate studies and for spending long hours going over this manuscript with me, taking care to understand definitions and proofs to their very bottom. I thank Larry Washington for being part of my introduction to cryptography and for continuing to be a part of my program ever since. I thank Dalit Naor and Amir Herzberg for their care and help throughout my graduate studies. I would also like to thank Fatima Bangura for helping me get through the maze that is the requirements of the degree, but more importantly for her care and friendship.

The work in this thesis was generously supported by three sources: the U.S. Air Force Research Laboratory, managing contract F30602-00-2-0510 for the Defense Advanced Research Projects Agency; the U.S. Army Research Office award DAAD19-01-1-0494; and the U.S. Army Research Laboratory and the U.K. Ministry of Defence under agreement W911NF-06-3-0001. I am grateful for this support.

I am fortunate to have found my best friend in graduate school. Gabe Rivera has been with me through it all, always there, supporting and making me feel safe in truly wanting only the best for me. This alone has been worth the trip. I thank my clone, Ronny Ever-Hadani, for his love and friendship. Ronny has inspired my path, ever since we discussed recursion during lunch breaks across from our army camp in Tel-Aviv, if you can call that a camp. I thank Georg Apitz for his enduring friendship, for being there when most people prefer not to, and for reminding me that I am here and it is now. Thanks to Chiu-Yuen Koo for being a true friend, for caring and for being the best crypto-partner one could hope for. I thank my cousin, Yonatan Bram, for his endless support and love. I thank Sarah Ellen Rasmussen for sharing part of the journey with me and bringing much happiness to it.

I thank Grecia Lapizco-Encinas for her love, her care, her patience and her endurance. Thank you for growing with me and opening my heart.

Finally, I thank my father Uri, my mother Zila and my sister Naama. Through successes and failures, joy and hardship, you have been the bedrock on which I relied. This could not, and would never have been completed if not for your unconditional love and support. This work is dedicated to you, Aba, Ima and Naama, with endless love.

Table of Contents

List of Figures

List of Abbreviations

IND-CPA	Indistinguishability under Chosen-Plaintext Attacks
WKA	Weak Key-Authenticity
WKA-EXP	Weak Key-Authenticity for Expressions
CF	Confusion Free
SKA	Strong Key-Authenticity
INT-PTXT	Integrity of Plaintext
PRG	Pseudorandom Generator
UOWHF	Universal One-Way Hash Function
PPT	Probabilistic, Polynomial Time
ITM	Interactive Turing Machine
UC	Universally-Composable
CRS	Common Reference String
SID	Session Identifier
DDH	Decisional Diffie-Hellman
OT	Oblivious Transfer
IBE	Identity-Based Encryption
RO	Random Oracle
RSA	Rivest Shamir Adleman
DCR	Decisional Composite Residuosity

Chapter 1

Introduction

Things derive their being and nature

by mutual dependence

and are nothing in themselves.

Acharya Nagarjuna, second century

The last three decades have seen cryptography approached from a computational-centric perspective. On the *definitional* side, the hallmark of the approach has been assuming little about the adversarial entity attacking the system beyond its computational abilities. In particular, definitions do not seek to address specific strategies that an adversary may employ (within the setting it is allowed to operate in); this is a desirable formulation, as an adversary will typically attempt to behave in a manner that was not envisioned during the design of the system. Symmetric encryption is a salient example — semantic security [GM84] requires that no probabilistic polynomial-time algorithm be able to distinguish an encryption of one message from an encryption of another with more than negligible probability of success. On the *constructive* side, modern solutions rely on, and indeed require, the ability to generate instances of problems that hard for computationally-bounded algorithms. Turning again to our example, we know that the existence of one-way functions — functions that are easy to compute but hard to invert for probabilistic, polynomial-time algorithms — is sufficient [BM84, GGM86, GL89, HILL99], and necessary [IL89], for constructing semantically-secure encryption schemes.

In examining the computational approach, it is important to consider the following questions:

- **Definitions:** How well do definitions capture the intended security goals?
- **Constructions:** Does the approach entail any inherent limitations on constructions that satisfy these definitions? If so, can we build protocols that match these limitations?

This thesis presents advances on both fronts. Towards one end, we look *outside* the framework and relate computational definitions with their counterparts in an alternative, prominent framework. Towards the other end, we look *inside* the framework and derive bounds on the efficiency of basic constructions. Below, we provide an informal, comprehensive account of our contributions.

1.1 What the Definitions Capture: Relating the Formal and Computational Treatments of Symmetric Encryption

A typical computational treatment [BDJR97, BKR00, BR93, BM84, GMW87, GM84, GMR88, Yao82] models cryptographic operations (or participants in a cryptographic protocol) as efficient algorithms on strings of bits. Security, as mentioned above, is defined in terms of probabilities of successful attacks by computationally-bounded adversaries. Such treatment offers concrete procedures for the notions it models and provides quantitative guarantees of security; it is, however, typically complex to model and argue in.

Cryptographic notions have received alternative treatments, the most prominent of which has been the *formal* one. Here, a typical treatment [AG99, BAN89, DY83, GM95, Kem89, KMM94, Low96, Mea91, MCF87, MMS97, Pau98, THG99] features a formal language, in which statements, representing cryptographic entities and operations, can be made. The security properties of these statements are usually asserted outside the language, or expressed as syntactic properties of the statements. The formal treatment is abstract, simple and lends itself promptly to automated analysis; it does not, however, offer concrete instantiations of the cryptographic notions it models. (Obviously, the lists of references mentioned above are not exhaustive.)

A significant effort has recently been made to relate the two approaches. Towards this end, researchers have been seeking conditions under which computational notions can be "plugged in" instead of the corresponding formal notions while preserving the formal semantics; and, conversely, conditions under which formal notions can abstract computational notions such that the computational semantics is maintained. For one, this offers a *direct means of seeing what aspects of the underlying natural concern were captured in each treatment*, shedding light on our topic of interest here. In addition, a central motivation has been the promise of bringing the strengths of one treatment to the other. Specifically, a successful endeavor is expected to confirm and increase the relevance of formal proofs to concrete computational instantiations; and allow the application of the high level formal reasoning mechanisms to the computational domain.

Related Work. Relating the formal and computational treatments of cryptography has been a thriving research area in recent years. Initial steps were taken by Abadi and Rogaway [AR02], building on the work of Aabadi and Jurgens [AJ01]; both deal with sufficient conditions under which formal semantics for symmetric encryption implies the computational semantics. Micciancio and Warinschi [MW04a] give a sufficient condition for the converse. Abadi and Warinschi [AW05] extend that work to relate the treatments of password-based encryption. Note that all the above deal with security notions in which the adversary plays a passive role.

A second line of research focuses on notions in which the adversary plays an active role. Micciancio and Warinschi [MW04b] formalize a setting in which the treatments of such notions can be related, and give a sufficient condition under which a symbolic notion of mutual authentication implies a computational one. Canetti and Herzog [CH04] lift these results to the Universal-Composability framework of Canetti [Can01], and deal additionally with the notion of key-exchange. Backes and Pfitzmann [BP05] relate formal and computational notions of secrecy in an equivalent framework of their design. Micciancio and Panjwani [MP05] extend the theorem of [AR02] to account for particular active cases and apply it to relate a formal treatment of multicast key-distribution with a computational one.

1.1.1 Our Contributions

We focus on relating the formal and computational treatments of symmetric encryption, completing the characterization initiated by Abadi and Rogaway [AR02]. We begin by reviewing their model and result, using a running example for intuition.

Consider a language of formal *expressions*, where expressions are built from symbols that represent bits and keys using symbolic operands that represent pairing and encryption. For example, the expression $E_1 = (K_1, \{0\}_{K_2})$ represents the pairing of key K_1 with an encryption of the bit 0 with key K_2. Two semantics are now defined. In the first, an expression is associated with a syntactic counterpart, the *pattern*, which mirrors the expression up to parts that should look "unintelligible" to an observer; informally, these are parts, defined in syntactic terms, that represent encryptions with keys that are not recoverable from the expression. In our example, E_1 will be mapped to the pattern $E_1 = (K_1, \Box)$, where $\Box$ represents an "unintelligible" part. Expressions are said to be *equivalent* in this setting if their patterns are equal as strings of symbols (up to a consistent renaming of the key symbols). For example, E_1 will be equivalent to $E_2 = (K_8, \{(1,1)\}_{K_9})$. This constitutes a *formal semantics.* In the second definition, an expression is associated with an ensemble of distributions on strings, obtained by *instantiating* the key symbols and encryption operations occurring in the expression with the corresponding components of a concrete computational encryption scheme (for all security parameters). Two expressions are said to be *indistinguishable* in this setting if their associated ensembles are computationally indistinguishable. This constitutes a *computational semantics.*

Within this setup, [AR02] give a soundness result: they show that under specific, sufficient conditions on the computational encryption scheme, equivalence of expressions (in the formal semantics) implies their indistinguishability (in the computational semantics). Our work tightly characterizes the completeness aspect of this exposition. We identify the following notion:

Definition 1.1.1 (The admittance of WKA-EXP tests for expressions - informal)**.** We say that a computational encryption scheme *admits a weak key-authenticity test for expressions* E_1, E_2 if there exists an efficient algorithm that distinguishes:

- an encryption of an instantiation of E_1, paired with the key used to perform the encryption; from
- an encryption of an instantiation of E_2, paired with a random key,

with a non-negligible probability (as a function of the security parameter used for the encryption scheme). Say that the scheme *admits weak key-authenticity tests for expressions* if it admits a weak key-authenticity test for *all* expressions E_1, E_2. $\diamondsuit$

As our main result, we show that:

Theorem 1.1.2 (Admittance of WKA-EXP tests is necessary and sufficient for formal encryption to be computationally-complete - informal)**.** *The admittance of weak key-authenticity tests is* necessary and sufficient *for indistinguishability of expressions (in the computational semantics) to imply their equivalence (in the formal semantics).*

In proving this result, we provide a novel fixpoint characterization of the syntactic notion of “intelligible” parts of formal expressions.

Prior to our result, a property of encryption schemes dubbed *confusion-freeness* was suggested as sufficient for completeness [AJ01, MW04a]. Roughly speaking, a confusion-free scheme is one in which the decryption of a ciphertext with a wrong key fails with almost certainty. The above-mentioned work suggests the use of a full-fledged *authenticated encryption scheme* [BN00, KY00] to provide for this property. As our second result, we show that:

Theorem 1.1.3 (Admittance of WKA-EXP is strictly weaker than related notions - informal)**.** *The requirement that an encryption scheme admits weak key-authenticity tests for expressions is* strictly weaker *than the requirement that it be confusion-free (and certainly weaker than the requirement that it be an authenticated encryption scheme).*

To that effect, we consider a strengthened version of our condition, requiring the admittance of a single, universal weak key-authenticity test (as opposed to one per pair of expressions), defined in strict computational terms (i.e., independent of expressions from the formal language). We say that an encryption scheme *admits a weak key authenticity test* if it satisfies this strengthened condition. We present a simple encryption scheme that admits a weak key-authenticity test but is *not* confusion-free. The scheme thus matches our completeness criterion, but not that of [MW04a]. Furthermore, it meets the soundness criterion of [AR02].

A preliminary version of this work appeared in [HG03].

1.2 Efficiency of Constructions: Bounds for Generic Commitments

Modern cryptography has had substantial success in identifying the minimal hardness assumptions needed for the construction of various cryptographic tools and protocols. We now know, for example, that one-way functions are necessary [IL89, Rom90] and sufficient for the construction of pseudorandom generators (PRGs) [BM84, Yao82, GL89, HILL99], universal one-way hash functions (UOWHFs) and digital signature schemes [NY89, Rom90], private-key encryption schemes [GGM84], and commitment schemes [Nao91]. It is the case, unfortunately, that all of the constructions just referenced are notoriously *inefficient*, and no constructions (based on one-way functions) improving upon the efficiency of these solutions are known. On the other hand, more efficient constructions are known to exist under stronger (e.g., number-theoretic) assumptions.

The apparent tradeoff between the efficiency of a construction and the underlying hardness assumption used to prove it secure has prompted a recent line of research aimed at answering the following question: *how efficient can constructions based on minimal assumptions be*? One way to formalize this question is to look at so-called "black-box" constructions which use an underlying primitive (e.g., a one-way permutation) only as an oracle, and which do not use, e.g., an explicit circuit computing the primitive in question (see Section 1.2.1 for further discussion). The idea of studying cryptographic constructions in this way was initiated by Impagliazzo and Rudich [IR89, Rud88] in the context of proving the impossibility of certain constructions, and much additional work in this vein followed

[Rud91, Sim98, GKM$^+$00, GMR01, Fis02]. (See [RTV04] for rigorous formal definitions of the Impagliazzo-Rudich model, as well as some variants that have been used.) Kim, Simon, and Tetali [KST99] were the first to use this model as a means of studying the *efficiency* of constructions (rather than their *feasibility*), with efficiency measured in terms of the number of oracle calls made by the construction. They show non-tight bounds on the efficiency of constructing UOWHFs from one-way permutations. Extending their results, Gennaro, et al. [GGKT05] show that known constructions of UOWHFs based on one-way permutations are in fact optimal; they also show efficiency bounds for the case of PRGs, private-key encryption schemes, and digital signatures based on one-way permutations, as well as for the case of public-key encryption schemes based on trapdoor permutations.

Before describing our contributions, we provide a brief overview of the Impagliazzo-Rudich model and black-box lower bounds. (The following is adapted from [GGKT05], including only what is directly relevant to the present work. For a more general discussion, see [GGKT05, RTV04].)

1.2.1 Black-Box Lower Bounds

At the most general level, a construction of a commitment scheme based on one-way permutations may be viewed as a procedure P which takes as input (a description of) a permutation π and outputs (a description of) two circuits $(\mathcal{S}, \mathcal{R})$ (here, $\mathcal{S}$ represents the *sender* while $\mathcal{R}$ represents the *receiver*; see Section 1.2.2 and Section 3.1.2) realizing the desired commitment functionality whenever π is a permutation. If the construction is *black-box*, this means that P relies only on the input/output behavior of π and *not* on any internal structure of the implementation of π; formally, this means that the construction can be described as a pair of oracle procedures $(\mathcal{S}^{(\cdot)}, \mathcal{R}^{(\cdot)})$ such that $(\mathcal{S}^{\pi}, \mathcal{R}^{\pi})$ realizes the desired functionality of a commitment scheme for any permutation π.

Besides achieving some desired functionality, a construction of a commitment scheme should also be "secure" in some sense. There are various ways this can be formalized (see [RTV04]); we will be interested here in *weak* black-box constructions which offer the following guarantee:

> If π is a *one-way* permutation, then the scheme $(\mathcal{S}^{\pi}, \mathcal{R}^{\pi})$ is "secure" against all efficient adversaries (who are *not* given oracle access to π),

where "secure" in the above refers to some appropriately-defined notions of hiding and binding. The distinction between whether an adversary is given oracle access to π or not is important since the above should hold *even when* π *is not efficiently computable* (and so the only way for an efficient adversary to evaluate π, in general, may be via oracle access to π). Note, however, that a weak black-box construction suffices to give implementations with meaningful security guarantees in the real world: in this case, π *will* be efficiently-computable and furthermore an explicit circuit for π will be known; hence, it is irrelevant whether an adversary is given oracle access to π or not. Note also that weak black-box constructions are the *weakest* type of black-box construction considered in [RTV04], and hence impossibility results for weak black-box constructions rule out other black-box constructions as well.

Although most currently-known constructions are black-box, it is important to recognize that a number of non-black-box constructions do exist. As an example, all known constructions of public-key encryption schemes secure against chosen-ciphertext attacks based on trapdoor permutations (e.g., [DDN00]) are non black-box. (See [GGKT05] for additional examples.) Nevertheless, a black-box impossibility result is useful in that it indicates the techniques necessary to achieve a particular result. Furthermore, known non-black-box constructions are much less efficient than black-box ones, and so a black-box impossibility result can be said to rule out "practical" constructions.

1.2.2 Our Contributions

With the above in mind, we may now describe our results in fairly formal terms. An interactive commitment scheme for m-bit messages is a pair of procedures $(\mathcal{S}, \mathcal{R})$ which operates in two phases. In the *commitment phase*, the *sender* $\mathcal{S}$ takes as input a message $M \in \{0,1\}^m$ and interacts with the *receiver* $\mathcal{R}$; we will refer to the view of $\mathcal{R}$ at the conclusion of this phase as a *commitment* to M. In the *decommitment phase*, the sender forwards a *decommitment* to $\mathcal{R}$ which, in particular, reveals M. Without loss of generality, we will assume that the decommitment simply consists of M along with the random coins used by $\mathcal{S}$ during the commitment phase.

A commitment scheme should guarantee both *hiding* and *binding*, where informally these mean that (1) the receiver should have no information about M at the end of the commitment phase while (2) the sender should be committed to a unique message at the end of that phase. More formally, a commitment scheme is *statistically-binding* if it satisfies the following:

Hiding: For any $M, M' \in \{0,1\}^m$, the distribution over commitments (by the honest sender $\mathcal{S}$) to M is computationally indistinguishable from the distribution over commitments (by $\mathcal{S}$) to M', even when $\mathcal{S}$ interacts with a malicious (but computationally bounded) receiver $\mathcal{R}^*$.

(Statistical) binding: The probability (over coin tosses of the honest receiver $\mathcal{R}$) that there exist distinct M, M' and coins s, s' for $\mathcal{S}$ such that the corresponding commitments to M, M' are identical is at most ε_b. When $\varepsilon_b = 0$ we say the scheme is *perfectly binding*.

Note that the formulation of the binding requirement ensures security even against an all-powerful sender. Our definition of the binding requirement is somewhat stronger than the usual one which, roughly speaking, requires only that a computationally-unbounded sender *without* knowledge of r be unable to *find* distinct M, M' and coins s, s' such that the corresponding commitments are identical (except with some probability ε_b). For the case of two-round, public-coin schemes (where the receiver simply sends a random string and the sender responds with a commitment) and perfectly-binding schemes, however, the notions are identical. Looking ahead, we remark that all the constructions we show in Section 3.3 satisfy the strong definition of binding given above.

Say a permutation $\pi : \{0,1\}^n \to \{0,1\}^n$ is *one-way with security S* if any circuit of size at most S inverts π on at most a fraction $1/S$ of its inputs. Our main result may be stated as follows:

Theorem 1.2.1 (Bounds on the efficiency of generic, statistically-binding commitments - informal)**.** *Any weak black-box construction of a statistically-binding commitment scheme based on a one-way permutation with security S requires*

$$\Omega\left((m - \log(1 + 2^m \cdot \varepsilon_b))/\log S\right)$$

invocations of the permutation (by the sender and receiver combined for statistically-binding schemes, and by the sender alone for perfectly-binding schemes).

Formally, we show that any construction beating this bound would imply the unconditional existence of a statistically-binding commitment scheme; or, put another way, the only way to develop a more efficient construction of a commitment scheme based on one-way permutations is to construct a commitment scheme *from scratch*. Note further that the existence of a commitment scheme implies the existence of one-way functions (and hence $\mathcal{P} \neq \mathcal{NP}$), and so in particular any black-box construction beating our bound would also imply a proof that $\mathcal{P} \neq \mathcal{NP}$. We remark that beyond the technical ideas used in our proof, our bound is interesting as the first example of an efficiency bound on a protocol which protects against malicious participants (the cryptographic primitives considered in [KST99, GGKT05] only involve honest participants, with the adversary being a "passive observer"; indeed, proving bounds for the case of commitment schemes was left as an explicit open question there).

For perfectly-binding schemes, our bound translates to $\Omega(m/\log S)$; our bound in this case matches the efficiency achieved by the construction of Blum [Blu82], instantiated using the Goldreich-Levin hard-core bits of a one-way permutation [GL89]. This is discussed further in Section 3.3, where we also compare our bounds to known constructions of statistically-binding schemes.

We remark that (a natural adaptation of) our bounds applies also to constructions of commitment schemes based on oracle access to trapdoor permutations (see [GGKT05] for definitions). For ease of exposition, however, we prefer to work with one-way permutations.

A preliminary version of this work appeared in [HK05].

1.3 Efficiency of Constructions: Universally Composable Two-Party Computation in Two Rounds

Round complexity is an important measure of efficiency for cryptographic protocols, and much research has focused on trying to characterize the round complexity of various tasks such as zero knowledge [GK96a, GK96b], concurrent zero knowledge [CKPR01, PRS02], Byzantine agreement [PSL80, FL82, FM97, GM98], Verifiable Secret-Sharing [GIKR01, FGG$^+$06], and secure two-party/multi-party computation [Yao86, BMR90, IK00, Lin01, GIKR02, KOS03, KO04]. (Needless to say, this list is not exhaustive.) We focus on the goal of secure two-party computation. Feasibility results in this case are clearly of theoretical importance, both in their own right and because two-party computation may be viewed as the "base case" for secure computation without honest majority. Results in this case are also of potential practical importance since many interesting cryptographic problems (zero knowledge, commitment, and — as we will see — blind signatures) can be solved by casting them as specific instances of secure two-party computation.

The round complexity of secure two-party computation in the *stand-alone* setting has been studied extensively. Yao [Yao86] gave the first constant-round protocol for the case when parties are assumed to be honest-but-curious. Goldreich, Micali, and Wigderson [GMW87, Gol04] showed how to obtain a protocol tolerating malicious adversaries; however, their protocol does not run in a constant number of rounds. Lindell [Lin01] gave the first constant-round protocol for secure two-party computation in the presence of malicious adversaries. Katz and Ostrovsky [KO04] showed a five-round protocol for malicious adversaries, and proved a lower bound showing that five rounds are necessary. (The lower bound applies to black-box proofs of security, and assumes parties communicate in alternate rounds.) Two-round protocols for secure two-party computation, where only a single player receives output, are studied in [SYY99, CCKM00]; in particular, Cachin et al. [CCKM00] show a two-round protocol for computing arbitrary, single-output functionalities assuming a *common reference string* (CRS) available to all participating parties.

It is by now well known that protocols secure when run in a stand-alone setting may no longer be secure when many copies of the protocol are run concurrently in an arbitrary manner (possibly among different parties), or when run alongside other protocols in a larger network. To address this issue, researchers have proposed models and definitions that would guarantee security in exactly such settings [PW00, Can01]. In this work, we adopt the model of *universal composability* (UC) introduced by Canetti [Can01].

The initial work of Canetti showed broad feasibility results for UC multi-party secure computation in the presence of a strict majority of honest players. Unfortunately, subsequent work of Canetti and Fischlin [CF01] showed that even for the case of two parties, one of whom may be malicious, there exist functionalities that cannot be securely computed within the UC framework; further characterization of all such "impossible-to-realize" two-party functionalities is given by [CKL06]. These impossibility results hold for the "plain" model; in contrast, it is known that these negative results can be bypassed if one is willing to assume some sort of "trusted setup" on which all parties can rely. Various forms of trusted setup have been explored [CF01, BCNP04, HMQU05, CDPW07, Katz07], the most common of which is the availability of a CRS to all parties in the network. Under this assumption, universally composable multi-party computation of any (well-formed) functionality is possible for any number of corrupted parties [CLOS02].

The round complexity of UC two-party computation has not been explored in detail. The two-party protocol given in [CLOS02] does not run in a constant number of rounds, though this may be due at least in part to the fact that the goal of their work was security under *adaptive* corruptions (where corruptions may happen at any point during the execution of the protocol, and not necessarily at its outset, as is the case with *passive* corruptions). Indeed, it is a long-standing open question to construct a constant-round protocol for adaptively-secure two-party computation even in the stand-alone setting. Jarecki and Shmatikov [JS07] recently showed a four-round protocol, assuming a CRS, for functionalities that generate output for only one of the parties; they also show a two-round protocol in the random oracle model. Using a standard transformation [Gol04], their protocols can be used to compute two-output functionalities at the cost of an additional round.

1.3.1 Our Contributions

As our main result here, we show the following:

Theorem 1.3.1 (UC two-party computation in two-rounds - informal)**.** *UC computation of any (well-formed) two-party functionality (where both parties may receive output) can be realized in only* two *rounds of communication, assuming static corruptions and the availability of a CRS to all participating parties.*

In our work, we allow both parties to simultaneously send a message in any given round (i.e., when both parties are honest), but prove security against a *rushing* adversary who may observe the other party's message in a given round before sending his own. Although this communication model is non-standard in the two-party setting, it matches the convention used in the study of multi-party protocols and allows for a more accurate characterization of the round complexity. Our result holds under any one of various standard number-theoretic assumptions, and does not rely on random oracles. We assume a CRS but, as we have seen, some form of setup is necessary for two-party computation to be possible (in any number of rounds). We consider static corruptions only; again, recall that even in the stand-alone setting it is not known how to achieve adaptive security in constant rounds.

We achieve our result via the following steps:

- We first show a two-round protocol (where only one party speaks in each round) for secure computation of any single-output functionality. This protocol is similar to that of Cachin et al. [CCKM00], though our protocol is secure in the UC framework. The protocol relies on Yao's "garbled circuit" technique [Yao86], the two-round oblivious transfer protocol of Tauman [Tau05], and the non-interactive zero-knowledge proofs of De Santis et al. [DDO+01]. Using standard techniques [Gol04, Propositions 7.2.11 and 7.4.4], this immediately implies a three-round protocol (where only one party speaks in each round) for any two-output functionality.

- As our main result, we show how two instances of our initial protocol can be run "in parallel" so as to obtain a two-round protocol (where now both parties speak[1] in each round) *even if both parties are to receive output*. The challenging aspect of this step is to "bind" the two executions to ensure that each party enters both executions with a consistent input.

It is not hard to see that one-round secure computation, even if both parties are allowed to speak simultaneously, is impossible under any reasonable definition of security and regardless of any global setup assumption; a similar observation holds for two-round protocols when parties speak in alternate rounds. Thus, interestingly, the round complexity of our protocols is optimal *for any setting of secure computation* and not "just" for the setting of universal composability with a CRS.

The low round complexity of our protocols implies round-efficient solutions for various cryptographic tasks. To give a timely example, we show that blind sig-

[1]We stress again that our security analysis takes into account a *rushing* adversary.

natures [Cha82] can be reduced to secure computation of a particular functionality (here, we simplify the prior result of [JLO97] to the same effect); thus, as almost an immediate corollary of our result we obtain a two-round blind signature protocol, matching a recent result by Fischlin [Fis06] via what is arguably a more intuitive construction. Our result also has certain technical advantages as compared to Fischlin's work: our scheme can be applied to *any* underlying signature scheme and achieves strong unforgeability "for free" (as long as the underlying signature scheme does); in contrast, Fischlin's result applies to a specific signature scheme of his design and achieves strong unforgeability only with significant additional complications. On the other hand, Fishlin's result holds under more general assumptions.

As a second example, we observe that the evaluation of a trust policy, held by a server, on a set of credentials, held by a client, can be cast as an instance of two-party computation. Applying our protocol yields a solution that provides input privacy to both the client and the server in a minimal number of rounds while preserving security under general composition, a combination of traits not present in current solutions (e.g., [BHS04, LDB03, NT05, LL06, BMC06, FAL06] and references therein).

A preliminary version of this work is to appear in [HK07].

Chapter 2

Relating the Formal and Computational Treatments of Symmetric Encryption

Here, we relate the formal and computational treatments of cryptography, as outlined in Section 1.1. The chapter is organized as follows. In Section 2.1, we describe the formal treatment of symmetric encryption, and provide a new procedural characterization of the formal semantics, to be used in the proof of our main theorem. In Section 2.2, we describe the computational treatment. In Section 2.3, we provide a necessary and sufficient condition for the computational semantics to imply the formal one. In Section 2.4, we demonstrate that our condition is strictly weaker than previously-suggested conditions, and present a simple, efficient construction.

2.1 Formal Treatment of Symmetric Encryption

Here, we present the formal view of symmetric encryption, most closely following its formalization in [AR02]. The view consists of a formal language and a formal semantics. Expressions in the formal language are built from symbols representing bits and keys, using operands that represent pairing and encryption. Formal semantics is defined in terms of equivalence of expressions, where equivalence is, in loose terms, syntactic identity up to parts representing encryptions with keys that cannot be recovered from expressions; we call such parts "unreachable". Our definitions recast those of [AR02] in terms that pertain closely to the syntactic structure of expressions. In addition, we provide a fixpoint characterization of the "reachable parts" of expressions that plays an important role in the proof of our main theorem.

2.1.1 Formal Language

2.1.1.1 Expressions

Let **Bits** be the set $\{0, 1\}$; we call $0, 1$ *bits*. Let **Keys** be a fixed, non-empty set of symbols, disjoint from **Bits**; call the elements of **Keys** *keys*. Our formal language consists of the set **Exp**, defined inductively as follows:

1. (**Atomic** elements or **Atoms**:) $\mathbf{Bits}, \mathbf{Keys} \subseteq \mathbf{Exp}$.

2. (**Non-Atomic** elements:)

   (a) **(The Pairing Rule:)** If $M, N \in \mathbf{Exp}$ then $(M, N) \in \mathbf{Exp}$. We say that (M, N) is *directly derived* from M and N.

   (b) **(The Encryption Rule:)** If $M \in \mathbf{Exp}$ and $K \in \mathbf{Keys}$, then $\{M\}_K \in \mathbf{Exp}$. We say that $\{M\}_K$ is *directly derived* from M.

Elements of **Exp** are called *expressions*. We refer to the pairing and the encryption rules as *derivation rules*. Informally and as expected, (M, N) represents the pairing of expressions M and N, while $\{M\}_K$ represents the encryption of expression M with key K.

Expressions are strings of symbols. The *length* of an expression E, denoted $|E|$, is the number of symbols it is comprised of (count '$\}_K$' as a single symbol).

We use $E_1 = E_2$ to denote that the expressions E_1, E_2 are identical as strings of symbols.

It is important to note that every non-atomic expression belongs in **Exp** by way of a unique derivation rule and a unique set of expressions it is derived from; we say that expressions in **Exp** are *uniquely readable* to express this property. Formally, we claim that for two non-atomic expressions E_1 and E_2, $E_1 = E_2$ if (and, clearly, only if) either:

- $E_1 = (M_1, N_1)$, $E_2 = (M_2, N_2)$ and $M_1 = M_2$, $N_1 = N_2$; or
- $E_1 = \{M_1\}_{K_1}$, $E_2 = \{M_2\}_{K_2}$ and $M_1 = M_2$, $K_1 = K_2$.

To see this, note first that non-atomic expressions always open and close with matching brackets (be it a '(','\)' pair or a '$\{$','$\}_K$' pair), and that expressions have an equal number of opening and closing brackets; these properties can easily be shown by induction on the structure of expressions. Also, note that only atomic expressions have length 1, and that no expression is shorter. Let $E_1 = E_2$. As the expressions are identical as strings of symbols, they have the same opening and closing brackets, and so must be in **Exp** by way of the same derivation rule. Assume $E_1 = (M_1, N_1) = (M_2, N_2) = E_2$ (the argument for $E_1 = \{M_1\}_{K_1} = \{M_2\}_{K_2} = E_2$ is similar). If $M_1 \neq M_2$, then one, say M_1, must be a proper prefix of the other. As $|M_1| \geq 1$, $|M_2| > 1$, and so M_2 must be non-atomic. But a proper prefix of a non-atomic expression, which closes with a bracket, does not have a balanced number of brackets and thus cannot be an expression—a contradiction. It follows that $M_1 = M_2$; similarly, $N_1 = N_2$.

2.1.1.2 Derivation Trees

The structure of an expression can be represented naturally in the form of a tree. A *derivation tree* T_E for an expression E is defined inductively as follows:

1. If E is atomic, then T_E consists of a single node, the *root*, labeled E.

2. If E is non-atomic, then T_E consists of a single node, the *root*, labeled E, and an ordered list of *trees* for the expressions from which E is directly derived; the sets of nodes of these trees are disjoint, and none contains the root of T_E. If $E = (M, N)$, we say that T_M and T_N are the *left* and *right subtrees* of T_E, respectively. The roots of T_M and T_N are said to be the *left* and *right children* of the root of T_E, respectively. Similarly, if $E = \{M\}_K$ then T_M is said to be the *subtree* of T_E; the root of T_M is said to be the *child* of the root of T_E.

Informally, the notion of a derivation tree resembles that of the standard parse tree; the two relate in that a node in a derivation tree is labeled with the *yield* of the corresponding node in the parse tree. We let $|T_E|$ denote the cardinality of the set of nodes of T_E.

We mention two properties of expressions and their derivation trees that are relevant to our treatment. First, two expressions are identical as strings of symbols iff their respective derivation trees are identical. For the *only if* part, argue inductively on the structure of expressions and rely on the unique-readability property of expressions; for the *if* part, argue inductively on the structure of derivation trees. Second, if $|E| = n$, then T_E consists of at most n nodes; this can be shown by induction on the length of expressions.

2.1.1.3 Acyclic Expressions

We consider expressions that contain no encryption cycles, following [AR02]. Say that a key K *appears in plain* in an expression E if there exists a node in T_E labeled K. We say that the key K_1 *encrypts* key K_2 in E if there exists a node v labeled $T_{\{M\}_{K_1}}$ in T_E (for some expression M) and there exists a node labeled K_2 is a subtree of v. This induces a relation on the keys that appear in plain in E, henceforth the *"encrypts"* relation. An expression E is said to be *acyclic* (respectively, *cyclic*) if its associated "encrypts" relation contains no cycles (respectively, contains cycles).

2.1.2 Formal Semantics

The semantics for a formal language involving the encryption operation aims at capturing privacy guarantees one expects the operation to provide. In particular, the semantics seeks to express our understanding that parts of expressions, representing encryptions with keys that are not recoverable from the text, should be unintelligible (or unreachable) to a viewer. Second, it seeks to capture the understanding that expressions, differing only in their unintelligible parts, should "look the same" to a viewer. This is done by mapping each expression to a syntactic counterpart, the *pattern*, which mirrors only its reachable parts; and by defining the equivalence of expressions in terms of the syntactic equality of their respective patterns (up to a consistent renaming of reachable keys).

2.1.2.1 Reachable Nodes

Let T_E be the derivation tree of an expression E, let V be its node set and let $r \in V$ be its root. A set $U \subseteq V$ is said to *contain the reachable nodes* of T_E if:

1. $r \in U$.

2. For all $u \in U$,

   (a) if u is labeled with an expression of the form (M, N), then both the children of u in T_E (labeled M and N) are in U.

   (b) if u is labeled with an expression of the form $\{M\}_K$, and there exists a $u' \in U$ labeled K, then the child of u in T_E (labeled M) is in U.

For example, the node set V of a derivation tree T_E for an expression E contains the reachable nodes of T_E.

For $E \in \mathbf{Exp}$ of length n, T_E consists of at most n nodes, of which there are at most 2^n subsets. It follows that the number of sets containing the set of reachable nodes of T_E is finite. Let R be the intersection of all those sets. It is easy to show that R itself contains the set of reachable nodes of T_E; it is minimal in the sense that it is contained in all such sets. We call R the *set of reachable nodes* of T_E; we call a node in R a *reachable* node. Informally, reachable nodes correspond to parts of an expression that should be intelligible to a viewer.

Let T_E be a derivation tree with root r and a set of reachable nodes R. The graph induced by T_E on R must be a tree rooted at r, and not a forest (otherwise, let R' be the set of nodes in the connected component that contains r; R' is a set that contains the set of reachable nodes in T_E, contradicting the minimality of R). We call this tree the *tree of reachable nodes*, and use T_E^R to denote it.

2.1.2.2 Patterns

The definition of a pattern extends that of an expression with the addition of an atomic symbol, $\Box$. Informally, $\Box$ will appear in parts of a pattern that correspond to unintelligible parts of the associated expression. Formally, Let **Pat** be a set, defined inductively as follows:

1. (**Atomic** elements or **Atoms**:) $\mathbf{Bits}, \mathbf{Keys}, \{\Box\} \subseteq \mathbf{Pat}$.
2. (**Non-Atomic** elements:)
   (a) **(The Pairing Rule:)** If $M, N \in \mathbf{Pat}$ then $(M, N) \in \mathbf{Pat}$. We say that (M, N) is *directly derived* from M and N.
   (b) **(The Encryption Rule:)** If $M \in \mathbf{Pat}$ and $K \in \mathbf{Keys}$, then $\{M\}_K \in \mathbf{Pat}$. We say that $\{M\}_K$ is *directly derived* from M.

We call the elements of **Pat** *patterns*. We define the length of a pattern and the equality of patterns as strings of symbols in a similar manner to the respective definitions for expressions. As there, we associate a derivation tree T_P with each pattern P, and can show that two patterns are identical as strings of symbols iff their respective derivation trees are identical.

To map expressions to patterns via their respective derivation trees, we will need an appropriate notion of tree isomorphism. Let T_1, T_2 be finite, rooted, ordered trees with node sets V_1, V_2 and roots $r_1 \in V_1$, $r_2 \in V_2$, respectively. T_1, T_2 are said to be *isomorphic as rooted, ordered trees* if there exists a bijection $\varphi : V_1 \to V_2$ such that:

1. $\varphi(r_1) = r_2$.
2. For all $v \in V_1$, $(u_1, \ldots, u_k)$ are the (ordered) children of v in T_1 iff $(\varphi(u_1), \ldots, \varphi(u_k))$ are the (ordered) children of $\varphi(v)$ in T_2.

φ is said to be an *isomorphism* of T_1, T_2 *as rooted, ordered trees.*

Let T_E be the derivation tree of an expression E, V_E its node set, $R \subseteq V_E$ its set of reachable nodes, and T_E^R the tree of reachable nodes of E. Let T_P be the derivation tree of a pattern P, V_P its node set. We say that *expression E has a pattern P* if there exists a $\varphi : R \to V_P$ such that:

1. φ is an isomorphism of T_E^R, T_P as rooted, ordered trees.
2. For all $v \in R$,
   (a) if v is labeled with a bit, then $\varphi(v)$ is labeled with an identical bit.
   (b) if v is labeled with a key, then $\varphi(v)$ is labeled with an identical key.
   (c) if v is labeled (M, N), then $\varphi(v)$ is labeled (M', N').
   (d) if v is labeled $\{M\}_K$ and there exists a $u \in R$ labeled K, then $\varphi(v)$ is labeled $\{M'\}_K$.
   (e) if v is labeled $\{M\}_K$ and there does not exist a $u \in R$ labeled K, then $\varphi(v)$ is labeled $\Box$.

The corresponding definition of [AR02] amounts to a walk of T_E^R and T_P that enforces the above constraints.

We note that the pattern P associated with each expression E is unique. To see this, notice that the uniqueness of T_E implies a unique set of reachable nodes R, which implies a unique T_E^R; T_E^R can then easily be shown to be mapped to a unique T_P under φ above, which, in turn, guarantees a unique P. The converse is not true, however; every pattern has infinitely many expressions that are mapped to it.

2.1.2.3 Equivalence

We proceed with the notion of expression equivalence. Informally, we require that the derivation trees of patterns corresponding to equivalent expressions be isomorphic up to key renaming. For $i \in \{1, 2\}$, let P_i be the pattern of expression E_i, with a derivation tree T_{P_i} over V_{P_i}. We say that E_1 is *equivalent* to E_2, and write $E_1 \cong E_2$, iff there exists a $\varphi : V_{P_1} \to V_{P_2}$ and a permutation σ on **Keys** such that:

1. φ is an isomorphism of T_{P_1}, T_{P_2} as rooted, ordered trees.

2. For all $v \in V_{P_1}$,

   (a) if v is labelled with a bit, then $\varphi(v)$ is labeled with an identical bit.

   (b) if v is labeled K, then $\varphi(v)$ is labeled with $\sigma(K)$.

   (c) if v is labeled (M, N), then $\varphi(v)$ is labeled (M', N').

   (d) if v is labeled $\{M\}_K$, then $\varphi(v)$ is labeled $\{M'\}_{\sigma(K)}$.

(e) if v is labeled $\Box$, then $\varphi(v)$ is labeled $\Box$.

Composing the definition of the pattern associated with an expression with the definition of expression equivalence, we obtain the following property of the equivalence relation.

Claim 2.1.1. *For $i \in \{1,2\}$, let E_i be an expression with a derivation tree T_{E_i}, a set of reachable nodes R_{E_i} and an induced tree of reachable nodes $T_{E_i}^{R_{E_i}}$. Then $E_1 \cong E_2$ iff there exist a $\varphi : R_{E_1} \to R_{E_2}$ and a permutation σ on **Keys** such that:*

1. *φ is an isomorphism of $T_{E_1}^{R_{E_1}}$, $T_{E_2}^{R_{E_2}}$ as rooted, ordered trees.*
2. *For all $v \in R_{E_1}$,*
   (a) *if v is labeled with a bit, then $\varphi(v)$ is labeled with an identical bit.*
   (b) *if v is labeled K, then $\varphi(v)$ is labeled with $\sigma(K)$.*
   (c) *if v is labeled (M,N), then $\varphi(v)$ is labeled (M',N').*
   (d) *if v is labeled $\{M\}_K$ and there exists a $u \in R_{E_1}$ labeled K, then $\varphi(v)$ is labeled $\{M'\}_{\sigma(K)}$ and $\varphi(u)$ is labeled $\sigma(K)$.*
   (e) *if v is labeled $\{M\}_K$ and there does not exist a $u \in R_{E_1}$ labeled K, then $\varphi(v)$ is labeled $\{M'\}_{K'}$ and there does not exist a $u' \in R_{E_2}$ labeled K'.*

We sketch the (mostly technical) proof here. For $i \in \{1,2\}$, let P_i be the pattern of E_i, and let $\varphi_i : R_{E_i} \to V_{P_i}$ be as in the definition of a pattern. For the *only if* part, let $\psi : V_{P_1} \to V_{P_2}$ and σ a permutation on **Keys**, witness the equivalence of E_1, E_2. Then $\varphi^* : R_{E_1} \to R_{E_2}$ such that $\varphi^*(v) = \varphi_2^{-1}(\psi(\varphi_1(v)))$ and $\sigma^* = \sigma$

are with properties as required in the claim. For the *if* part, let $\varphi^* : R_{E_1} \to R_{E_2}$ and σ a permutation on **Keys** be with properties as specified in the claim. Then $\psi : V_{P_1} \to V_{P_2}$ such that $\psi(v) = \varphi_2(\varphi^*(\varphi_1^{-1}(v)))$ and $\sigma = \sigma^*$ witness the equivalence of E_1, E_2.

2.1.2.4 Properties captured by the formal semantics

We conclude with a brief discussion of some ramifications of the formal semantics we have seen in this section. Our intention is twofold: to point out definitional choices made in the formal setting, and to highlight aspects that need to be addressed in the computational setting as well. We observe that under the above definitions, the encryption operator seeks to:

- *"Preserve data privacy"*, as seen in the equivalence $\{0\}_K \cong \{1\}_K$. Informally, a ciphertext conceals the underlying plaintext.

- *"Conceal plaintext repetitions"*, as seen in the equivalence $(\{0\}_K, \{0\}_K) \cong (\{0\}_K, \{1\}_K)$. Informally, an adversary, given two ciphertexts, cannot tell whether their underlying plaintexts are identical or not.

- *"Conceal key repetitions"*, as seen in the equivalence $(\{0\}_{K_1}, \{1\}_{K_1}) \cong (\{0\}_{K_7}, \{1\}_{K_8})$. Informally, an adversary, given two ciphertexts, cannot tell whether they were generated with the same encryption key or not.

- *"Conceal plaintext length"*, as seen in the equivalence $\{0\}_K \cong \{(0,(1,0))\}_K$. Informally, the ciphertext conceals the length of the underlying plaintext.

The definitions of the semantics can be modified to accommodate relaxations of the above properties. For example, the semantics can be made sensitive to different plaintext lengths, by introducing an atomic pattern symbol $\Box_n$ for each size n and modifying the definition of equivalence appropriately. We stress that the results of [AR02] and ours can be modified to tolerate such changes.

2.1.3 A Fixpoint Characterization of the Set of Reachable Nodes

The set of reachable nodes of T_E for some expression E was defined in Section 2.1.2.1 in set-intersection terms. Here, we characterize that set in terms of the least-fixpoint of an associated operator, O_E. Moreover, we show that the fixpoint can be computed by a applying the operator to its own output a number of times at most $|E|$.

Let S be a finite set, and let 2^S be the set of all subsets of S. A function $O : 2^S \to 2^S$ is said to be *monotonic* if for all $A, B \in 2^S$, $A \subseteq B$ implies $O(A) \subseteq O(B)$. A set $A \subseteq 2^S$ is said to be a *fixpoint* of $O : 2^S \to 2^S$ if $O(A) = A$; A is said to be the *least fixpoint* of O if A is a fixpoint of O and for all fixpoints B of O, $A \subseteq B$.

The following is a weak form of a theorem due to Tarski [Tar55]. For a general treatment of Lemma 2.1.2 and Lemma 2.1.3 see [Llo87, Section 1.5]; for an interesting account of related results, see [LN84].

Lemma 2.1.2. *Let S be a finite set, $O : 2^S \to 2^S$ a monotonic function. Then O has a least fixpoint, denoted* $\mathrm{lfp}(O)$*. Furthermore,*

$$\mathrm{lfp}(O) = \bigcap \left\{ X | X \in 2^S, O(X) \subseteq X \right\}.$$

Proof. Let $A = \bigcap \{X | X \in 2^S, O(X) \subseteq X\}$.

We first show that $O(A) \subseteq A$. Let $X \in 2^S$ be such that $O(X) \subseteq X$. Clearly $A \subseteq X$. By the monotonicity of O, $O(A) \subseteq O(X)$, and so $O(A) \subseteq X$. It follows that $O(A) \subseteq \bigcap \{X | X \in 2^S, O(X) \subseteq X\} = A$.

Next, we show that $A \subseteq O(A)$. We just proved that $O(A) \subseteq A$. By the monotonicity of O, $O(O(A)) \subseteq O(A)$. Therefore $O(A) \in \{X | X \in 2^S, O(X) \subseteq X\}$. By the definition of A, it follows that $A \subseteq O(A)$.

Putting it all together, we conclude that A is a fixpoint of O.

Let $B \in 2^S$ be any fixpoint of O. Then $B \in \{X | X \in 2^S, O(X) \subseteq X\}$. By the definition of A, it follows that $A \subseteq B$. A is thus the least fixpoint of O. ■

Let S be a finite set, $O : 2^S \to 2^S$ a monotonic function. The *powers* of O are defined inductively as follows:

$$O^0 = \emptyset$$

$$O^i = O(O^{i-1}) \quad \text{for all } i \in \mathbb{N}^+$$

We give a characterization of $\text{lfp}(O)$ in terms of the powers of O.

Lemma 2.1.3. *Let S be a finite set, $O : 2^S \to 2^S$ a monotonic function. Then there exists an $i \leq |S|$ such that for all $j \geq i$, $O^j = \text{lfp}(O)$.*

Proof. First note that for all i,

$$O^i \subseteq \text{lfp}(O). \tag{2.1}$$

This can be shown by induction: clearly, $O^0 = \emptyset \subseteq \text{lfp}(O)$; as for the step, $O^i =$

$O(O^{i-1}) \subseteq O(\text{lfp}(O)) \subseteq \text{lfp}(O)$ by the induction hypothesis, the monotonicity of O, and the definition of a fixpoint of O.

Next, observe that for all i,

$$O^i \subseteq O^{i+1}. \tag{2.2}$$

This can be shown again by induction: $O^0 = \emptyset \subseteq O^1$; $O^i = O(O^{i-1}) \subseteq O(O^i) = O^{i+1}$, by the induction hypothesis and the monotonicity of O. Using this property, we can now show that for all $j \geq i$,

$$O^i \subseteq O^j. \tag{2.3}$$

This, by induction on j: clearly, for $j = i$, $O^i \subseteq O^j$; furthermore, $O^i \subseteq O^{j-1} \subseteq O^j$ by the induction hypothesis and Equation 2.2.

Now note that for all i,

$$\text{if } O^i = O^{i+1}, \text{ then } O^i = \text{lfp}(O). \tag{2.4}$$

To see this, observe that O^i is a fixpoint of O because $O^i = O^{i+1} = O(O^i)$, and so $\text{lfp}(O) \subseteq O^i$ by the definition of a least fixpoint. But by Equation 2.1 we know that for all i, $O^i \subseteq \text{lfp}(O)$. It follows that $O^i = \text{lfp}(O)$.

Assume towards a contradiction that there does not exist an i, $0 \leq i \leq |S|$, such that $O^i = O^{i+1}$. Then we can use induction to show that for all $0 \leq k \leq |S|+1$, $\left|O^k\right| \geq k$. For the base case, observe that $|O^0| = |\emptyset| = 0$. For the step, note that $O^{k-1} \neq O^k$ by our assumption, and that $O^{k-1} \subseteq O^k$ by Equation 2.2. It follows that $\left|O^k\right| \geq \left|O^{k-1}\right| + 1$. By the induction hypothesis, we have that $\left|O^{k-1}\right| \geq (k-1)$, and so $\left|O^k\right| \geq k$, completing the induction step. But then $\left|O^{|S|+1}\right| \geq |S| + 1$ — a contradiction, as S contains only $|S|$ elements.

We conclude that there exists an i, $0 \leq i \leq |S|$, such that $O^i = O^{i+1}$. By Equation 2.4, we thus have that

$$O^i = \mathrm{lfp}(O). \tag{2.5}$$

Let j be such that $j \geq i$. Then $O^i \subseteq O^j \subseteq \mathrm{lfp}(O) = O^i$ by Equation 2.3, Equation 2.1 and Equation 2.5, respectively. It follows that $O^j = \mathrm{lfp}(O)$. ■

Let E be an expression, T_E its derivation tree over a set of nodes V_E with a root $r_E \in V_E$. Let $O_E : 2^{V_E} \to 2^{V_E}$ be defined as follows:

$$O_E(A) = \left\{ u \in V_E \;\middle|\; \begin{array}{ll} \multicolumn{2}{l}{\text{either:}} \\ \text{(a)} & u = r_E\text{; or} \\ \text{(b)} & \exists v \in A \text{ labeled } (M, N) \text{ with a left child } u \text{ in } T_E \text{ (labeled } M\text{); or} \\ \text{(c)} & \exists v \in A \text{ labeled } (M, N) \text{ with a right child } u \text{ in } T_E \text{ (labeled } N\text{); or} \\ \text{(d)} & \exists v \in A \text{ labeled } \{M\}_K \text{ with a child } u \text{ in } T_E \text{ (labeled } M\text{)} \\ & \text{and } \exists w \in A \text{ labeled } K. \end{array} \right\}$$

It is easy to show that O_E is monotonic over sets in 2^S. Furthermore, we have the following:

Lemma 2.1.4. *For all $A \in 2^{V_E}$, $O_E(A) \subseteq A$ iff A is a set that contains the set of reachable nodes of T_E.*

Proof. Assume $O_E(A) \subseteq A$. First observe that $r_E \in O_E(A) \subseteq A$, and so $r_E \in A$. Furthermore, if $v \in A$ is labeled (M, N) and has a child u in T_E, then $u \in O_E(A) \subseteq$

A, and so $u \in A$. Similarly, if $v \in A$ is labeled $\{M\}_K$, has a child u in T_E and $\exists w \in A$ labeled K, then $u \in O_E(A) \subseteq A$, and so $u \in A$. It follows that A is a set that contains the set of reachable nodes of T_E.

Conversely, assume A is such a set, and let u be an element of $O_E(A)$. Then either:

- $u = r_E$. Clearly $u \in A$.
- $\exists v \in A$ labeled (M, N) with a child u in T_E. As A contains the reachable nodes of T_E, $u \in A$.
- $\exists v \in A$ labeled $\{M\}_K$ with a child u in T_E, $\exists w \in A$ labeled K. Again, A contains the set of reachable nodes of T_E, and so $u \in A$.

It follows that $O_E(A) \subseteq A$. ■

Putting it all together:

Theorem 2.1.5 (A Fixpoint Characterization of the Set of Reachable Nodes)**.** *Let E be an expression of length n, T_E its derivation tree over V_E, $R_E \subseteq V_E$ the set of reachable nodes. Then there exists an $i \in \mathbb{N}$, $0 \leq i \leq n$, such that for all $j \geq i$, $O_E^j = \mathrm{lfp}(O_E) = R_E$.*

Proof. Recall that $|V_E| \leq n$ (see Section 2.1.1.2) and that $O_E : 2^S \to 2^S$ is monotonic; we have that:

$$\begin{aligned}
R_E &= \bigcap \left\{ A \,\middle|\, A \in 2^{V_E}, \begin{array}{c} A \text{ contains the set of} \\ \text{reachable nodes of } T_E \end{array} \right\} && \text{by def. of } R_E \\
&= \bigcap \{ A \,|\, A \in 2^{V_E}, O_E(A) \subseteq A \} && \text{by lemma 2.1.4} \\
&= \text{lfp}(O) && \text{by lemma 2.1.2} \\
&= O_E^j && \text{by lemma 2.1.3,}
\end{aligned}$$

for any j and some particular i such that $0 \leq i \leq |V_E| \leq n$ and $j \geq i$. ■

2.2 Computational Treatment of Symmetric Encryption

Here, we overview the computational treatment of symmetric encryption, following the formalization in [AR02] (for ease of reading our results in the context of theirs). We begin by defining a computational encryption scheme, discuss a relevant notion of security, and review methods of achieving such a notion under standard assumptions. We then define computational "instantiations" of expressions, obtained by replacing keys and the encryption operator in expressions with their counterparts from an encryption scheme. We define semantics in terms of the computational indistinguishability of the instantiated expressions. Our definition of computational instantiations of expressions recasts that of [AR02] in terms of the derivation trees of the expressions.

We remark that notions of computational indistinguishability defined in this section and used in this chapter all consider probabilistic, polynomial-time adversaries. Contrast with indistinguishability by polynomial-sized circuit families, used in Chapter 3, and with indistinguishability by probabilistic, polynomial-time algorithms taking non-uniform "advice", used in Chapter 4; see discussion in Section 4.1.1. Throughout this section, recall that a function $\epsilon : \mathbb{N} \to \mathbb{R}$ is *negligible* if for every constant $c \in \mathbb{N}$ there exists an $\eta_c \in \mathbb{N}$ such that for all $\eta \in \mathbb{N}$ such that $\eta > \eta_c$, $\epsilon(\eta) \leq \eta^{-c}$.

2.2.1 Computational Encryption, Security and Tools

2.2.1.1 Encryption Schemes

Let $\{0,1\}^*$ denote the set of all finite binary strings and let $|x|$ denote the length of $x \in \{0,1\}^*$. An *encryption scheme* $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ with a *security parameter* $\eta \in \mathbb{N}$ consists of three polynomial-time algorithms, as follows:

- $\mathcal{K}$, the *key generation algorithm*, is a probabilistic algorithm that takes a security parameter $\eta \in \mathbb{N}$ (provided in unary—denoted by 1^η) and returns a *key* $k \in \{0,1\}^*$. We write $k \overset{R}{\leftarrow} \mathcal{K}(1^\eta)$, thinking of k as being drawn from the probability distribution induced by $\mathcal{K}(1^\eta)$ on $\{0,1\}^*$. When used as a set, we let $\mathcal{K}(1^\eta)$ denote the support of that distribution. For ease of exposition, we make the simple assumption that $\mathcal{K}$ *distributes keys decently*, that is, $\Pr[k, k' \overset{R}{\leftarrow} \mathcal{K}(1^\eta) : k \neq k']$ is non-negligible (as a function of η).

- $\mathcal{E}$, the *encryption algorithm*, is a probabilistic algorithm that takes a key $k \in \mathcal{K}(1^\eta)$ for some $\eta \in \mathbb{N}$ and a *plaintext* $x \in \{0,1\}^*$ and returns a *ciphertext* $c \in \{0,1\}^* \cup \{\bot\}$. As before, we write $c \overset{R}{\leftarrow} \mathcal{E}_k(x)$, thinking of c as being drawn from the probability distribution induced by $\mathcal{E}_k(x)$ on $\{0,1\}^*$. When used as a set, we let $\mathcal{E}_k(x)$ denote the support of that distribution.

It is common for encryption schemes to restrict the set of strings they are willing to encrypt; having the encryption algorithm return $\perp$ is intended to allow capturing such restrictions. Call a plaintext $x \in \{0,1\}^*$ *restricted* for some $\eta \in \mathbb{N}$ if for all $k \in \mathcal{K}(1^\eta)$, $\mathcal{E}_k(x) = \{\perp\}$. Call $x \in \{0,1\}^*$ *unrestricted* for $\eta \in \mathbb{N}$ if for all $k \in \mathcal{K}(1^\eta)$, $\mathcal{E}_k(x) \not\ni \perp$. We require that x is either restricted or unrestricted for any given η. Use $\text{Plain}_{\Pi[\eta]}$ to denote the set of unrestricted plaintexts for any $\eta \in \mathbb{N}$. We further require that for any $\eta \in \mathbb{N}$, if $x \in \{0,1\}^*$ is unrestricted for η, then all $x' \in \{0,1\}^*$ such that $|x'| = |x|$ are unrestricted for η.

In addition, we insist that the length of a ciphertext $c \in \mathcal{E}_k(x)$ depend only on η and $|x|$ when $k \in \mathcal{K}(1^\eta)$, for any x and η.

- $\mathcal{D}$, the *decryption algorithm*, is a deterministic algorithm that takes a key $k \in \mathcal{K}(1^\eta)$ for some $\eta \in \mathbb{N}$ and a ciphertext $c \in \{0,1\}^*$ and returns some $x \in \{0,1\}^* \cup \{\perp\}$. We write $x \leftarrow \mathcal{D}_k(c)$.

  Having the decryption algorithm output $\perp$ is intended to reflect a rejection of the given ciphertext.

We require that Π be *correct*; that is, for all $\eta \in \mathbb{N}$, for all $k \in \mathcal{K}(1^\eta)$ and for all $x \in \text{Plain}_{\Pi[\eta]}$, $\mathcal{D}_k(\mathcal{E}_k(x)) = x$.

2.2.1.2 Type-0 Security

We consider a notion of security for symmetric encryption that is variation on the standard notion of indistinguishability under chosen-plaintext attacks (*IND-CPA* security, for short) of [GM84, BDJR97], following [AR02]. Informally, and recalling our discussion of Section 2.1.2.4, the standard notion "preserves privacy" and "conceals plaintext repetitions", while the strengthened version "conceals key repetitions" and "conceals message lengths" as well; the strengthening is necessary for proving that the formal semantics is computationally *sound* (see [AR02] for discussion).

Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an encryption scheme, $\eta \in \mathbb{N}$ a security parameter and A an adversary with access to two oracles (denoted $A^{(\cdot),(\cdot)}$). Define:

$$\begin{aligned}\mathrm{Adv}^{0}_{\Pi[\eta]}(A) = & \Pr[k, k' \overset{R}{\leftarrow} \mathcal{K}(1^\eta) : A^{\mathcal{E}_k(\cdot),\mathcal{E}_{k'}(\cdot)}(1^\eta) = 1] \\ & - \Pr[k \overset{R}{\leftarrow} \mathcal{K}(1^\eta) : A^{\mathcal{E}_k(0),\mathcal{E}_k(0)}(1^\eta) = 1],\end{aligned}$$

where $\mathcal{E}_k(\cdot)$ is an oracle that returns $c \overset{R}{\leftarrow} \mathcal{E}_k(m)$ on input m, and $\mathcal{E}_k(0)$ is an oracle that returns $c \overset{R}{\leftarrow} \mathcal{E}_k(0)$ on input m. We say that Π is *Type-0 secure* (or *IND-CPA, Key-repetition Concealing, Length Concealing secure*) [AR02] if for every probabilistic, polynomial-time adversary A, $\mathrm{Adv}^{0}_{\Pi[\eta]}(A)$ is negligible (as a function of η).

2.2.1.3 Pseudorandom Function Families and Obtaining Type-0 Security

Let $x \overset{R}{\leftarrow} S$ denote the sampling of x from a set S under the uniform distribution. Let $\eta \in \mathbb{N}$, l, L be polynomials, $\mathrm{Func}^{l(\eta)\to L(\eta)}$ the set of all functions from

$\{0,1\}^{l(n)}$ to $\{0,1\}^{L(n)}$, $F \subseteq \mathrm{Func}^{l(\eta)\to L(\eta)}$ a family of functions indexed by $\{0,1\}^\eta$, and A an adversary with access to an oracle (denoted $A^{(\cdot)}$). Define:

$$\begin{aligned}\mathrm{Adv}^{\mathrm{prf}}_{F[\eta]}(A) = &\Pr[k \xleftarrow{R} \{0,1\}^\eta : A^{F_k(\cdot)}(1^\eta) = 1]\\ &- \Pr[f \xleftarrow{R} \mathrm{Func}^{l(\eta)\to L(\eta)} : A^{f(\cdot)}(1^\eta) = 1],\end{aligned}$$

where $F_k(\cdot)$ is an oracle that returns $F_k(x)$ on input x, and $f(\cdot)$ is an oracle that returns $f(x)$ on input x. We say that F is *pseudorandom* [GGM86] if for every probabilistic, polynomial time adversary A, $\mathrm{Adv}^{\mathrm{prf}}_{F[\eta]}(A)$ is negligible (as a function of η).

Bellare et. al. [BDJR97] show that the CBC and CTR modes of encryption with underlying pseudorandom function families are IND-CPA secure. For a description of how these results extend to achieve type-0 security, see [AR02].

2.2.2 Computational Semantics

Here, we define a computational semantics for the language of expressions of Section 2.1.1. We first associate with each expression an ensemble of distributions over bit-strings; each distribution is obtained by "instantiating" the expression with keys and an encryption operation provided by a computational encryption scheme, using a particular security parameter. We then define expression indistinguishability in terms of the computational indistinguishability of the associated ensembles.

2.2.2.1 Instantiating Expressions

Let E be an expression and let T_E be its derivation tree over V_E. Say that a key K *appears* in E if there exists a node in T_E labeled K or $\{M\}_K$ (for some expression M). Let Keys_E be the set of key symbols appearing in E. Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an encryption scheme with a security parameter $\eta \in \mathbb{N}$. For $x_1, \ldots, x_k \in \{0,1\}^*$ and a tag t from some finite, fixed set of tags, let $\langle x_1, \ldots, x_k, t\rangle$ denote an (arbitrary, fixed, unambiguous, polynomial-time) encoding of $x_1, \ldots, x_k, t$ as a string over $\{0,1\}^*$. Define the following procedure:

$\mathbf{Sample}_{\mathbf{\Pi}[\eta]}(\mathbf{E})$

1. For each $K \in \text{Keys}_E$, let $\tau(K) \overset{R}{\leftarrow} \mathcal{K}(1^\eta)$.
2. Assign a *sampling label* to each $v \in V_E$, inductively, as follows:
   (a) If v is labeled with a bit b, let its *sampling label* be $\langle b, \text{“bit”}\rangle$.
   (b) If v is labeled with a key K, let its *sampling label* be $\langle \tau(K), \text{“key”}\rangle$.
   (c) If v is labeled (M, N), its left child in T_E has a sampling label m and its right child in T_E has a sampling label n, then let the *sampling label* of v be $\langle m, n, \text{“pair”}\rangle$ if $m, n \neq \perp$, $\perp$ otherwise.
   (d) If v is labeled $\{M\}_K$ and its child in T_E has a sampling label m, then let the *sampling label* of v be $\langle \mathcal{E}_{\tau(K)}(m), \text{“ciphertext”}\rangle$ if $m \neq \perp$, $\perp$ otherwise.
3. Output the sampling label of the root of T_E.

Let $[\![E]\!]_{\Pi(\eta)}$ denote the probability distribution induced by $\mathsf{Sample}_{\Pi[\eta]}(E)$ on $\{0,1\}^* \cup \perp$; let $[\![E]\!]_\Pi$ denote the ensemble $\left\{[\![E]\!]_{\Pi(\eta)}\right\}_{\eta\in\mathbb{N}}$. We write $x \overset{R}{\leftarrow} D$ to indicate that x is sampled from a distribution D. To make our forthcoming definitions robust, we require that Π is such that for every expression E, there exists an $\eta_E \in \mathbb{N}$ such that for all $\eta \geq \eta_E$ and $e \overset{R}{\leftarrow} [\![E]\!]_{\Pi(\eta)}$, $e \in \mathrm{Plain}_{\Pi[\eta]}$.

2.2.2.2 Indistinguishability

For $i \in \{1, 2\}$, let $D_i = \{D_i(\eta)\}_{\eta\in\mathbb{N}}$ be probability distribution ensembles, A an algorithm. Define:

$$\begin{aligned}\mathrm{Adv}^{\mathrm{ind}}_{D_1(\eta),D_2(\eta)}(A) = &\Pr[x \overset{R}{\leftarrow} D_1(\eta) : A(1^\eta, x) = 1] \\ &- \Pr[x \overset{R}{\leftarrow} D_2(\eta) : A(1^\eta, x) = 1].\end{aligned}$$

We say that D_1, D_2 are *indistinguishable*, and write $D_1 \overset{\mathrm{c}}{\approx} D_2$, if for every probabilistic, polynomial time algorithm A, $\mathrm{Adv}^{\mathrm{ind}}_{D_1(\eta),D_2(\eta)}(A)$ is negligible (as a function of η). (Once again, see Section 4.1.1 for comparison with notions of indistinguishability used in the other chapters.)

Let E_1, E_2 be expressions. We say that E_1, E_2 are *indistinguishable*, and write $E_1 \overset{\Pi}{\approx} E_2$, iff $[\![E_1]\!]_\Pi \overset{\mathrm{c}}{\approx} [\![E_2]\!]_\Pi$.

2.3 Relating the Treatments – Formal Encryption is Computationally Complete

The soundness result of Abadi and Rogaway states that for acyclic expressions E_1, E_2 and a Type-0 encryption scheme Π, $E_1 \cong E_2$ implies $E_1 \overset{\Pi}{\approx} E_2$. Here, we give a *necessary* and *sufficient* condition for the converse, thereby tightly characterizing the completeness aspect of the exposition. For any two acyclic expressions, the condition involves the admittance of an efficient test that distinguishes a ciphertext and the key it was encrypted with, from a ciphertext and some random key, with a non-negligible probability, when the plaintexts are drawn from the ensembles associated with those expressions. Formally:

Definition 2.3.1 (Weak Key-Authenticity Test/s for Expressions)**.** Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an encryption scheme with a security parameter $\eta \in \mathbb{N}$, let E_1, E_2 be acyclic expressions, A an algorithm. Define:

$$\begin{aligned}
&\mathrm{Adv}^{\text{wka-exp}}_{\Pi[\eta],E_1,E_2}(A)\\
&\qquad = \Pr[e \overset{R}{\leftarrow} [\![E_1]\!]_{\Pi(\eta)}; k \overset{R}{\leftarrow} \mathcal{K}(1^\eta); c \overset{R}{\leftarrow} \mathcal{E}_k(e) : A(1^\eta, c, k) = 1]\\
&\qquad\quad - \Pr[e \overset{R}{\leftarrow} [\![E_2]\!]_{\Pi(\eta)}; k, k' \overset{R}{\leftarrow} \mathcal{K}(1^\eta); c \overset{R}{\leftarrow} \mathcal{E}_k(e) : A(1^\eta, c, k') = 1].
\end{aligned}$$

We say that Π *admits a weak key-authenticity test for* E_1, E_2 (*WKA-EXP-*(E_1, E_2) *test*, for short), if there exists a probabilistic, polynomial-time algorithm A such that $\mathrm{Adv}^{\text{wka-exp}}_{\Pi[\eta],E_1,E_2}(A)$ is non-negligible (as a function of η).

We say that Π *admits weak key-authenticity tests for expressions* (*WKA-EXP tests*, for short), if for all acyclic expressions E_1 and E_2, Π admits a weak key-authenticity test for E_1, E_2. ♢

Our main result is the following:

Theorem 2.3.2 (The admittance of WKA-EXP tests is necessary and sufficient for formal encryption to be computationally-complete)**.** *Let* $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ *be an encryption scheme. Then for all acyclic expressions* E_1 *and* E_2*,* $E_1 \overset{\Pi}{\approx} E_2$ *implies* $E_1 \cong E_2$ iff Π *admits weak key-authenticity tests for expressions.*

We proceed with the proof below.

2.3.1 Admittance of WKA-EXP Tests is Necessary for Formal Encryption to be Computationally-Complete

Here, we prove the *only-if* part of Theorem 2.3.2. Let E_1, E_2 be two acyclic expressions. Consider the expressions $M_1 = (\{E_1\}_K, K)$, $M_2 = (\{E_2\}_K, K')$ (without loss of generality, assume K does not appear in E_1, E_2). $M_1 \not\cong M_2$, so by the completeness assumption, we have that $M_1 \overset{\Pi}{\not\approx} M_2$. Let B be such that $\mathrm{Adv}^{\mathrm{ind}}_{[\![M_1]\!]_{\Pi(\eta)},[\![M_2]\!]_{\Pi(\eta)}}(B)$ is non-negligible. We use B to construct a WKA-EXP-(E_1, E_2) test A for Π. Define:

$$A(1^\eta, c, k) \overset{\mathrm{def}}{=} B(1^\eta, \langle\langle c, \text{“ciphertext”}\rangle, \langle k, \text{“key”}\rangle, \text{“pair”}\rangle).$$

We have:

$$\begin{aligned}
&\mathrm{Adv}^{\text{wka-exp}}_{\Pi[\eta],E_1,E_2}(A)\\
&\quad= \Pr[e \overset{R}{\leftarrow} [\![E_1]\!]_{\Pi(\eta)}; k \overset{R}{\leftarrow} \mathcal{K}(1^\eta); c \overset{R}{\leftarrow} \mathcal{E}_k(e) : A(1^\eta, c, k) = 1]\\
&\qquad - \Pr[e \overset{R}{\leftarrow} [\![E_2]\!]_{\Pi(\eta)}; k, k' \overset{R}{\leftarrow} \mathcal{K}(1^\eta); c \overset{R}{\leftarrow} \mathcal{E}_k(e) : A(1^\eta, c, k') = 1]\\
&\quad= \Pr\left[e \overset{R}{\leftarrow} [\![E_1]\!]_{\Pi(\eta)}; k \overset{R}{\leftarrow} \mathcal{K}(1^\eta); c \overset{R}{\leftarrow} \mathcal{E}_k(e) : B\left(1^\eta, \begin{array}{c}\langle\langle c, \text{“ciphertext”}\rangle,\\ \langle k, \text{“key”}\rangle, \text{“pair”}\rangle\end{array}\right) = 1\right]\\
&\qquad - \Pr\left[e \overset{R}{\leftarrow} [\![E_2]\!]_{\Pi(\eta)}; k, k' \overset{R}{\leftarrow} \mathcal{K}(1^\eta); c \overset{R}{\leftarrow} \mathcal{E}_k(e) : B\left(1^\eta, \begin{array}{c}\langle\langle c, \text{“ciphertext”}\rangle,\\ \langle k', \text{“key”}\rangle, \text{“pair”}\rangle\end{array}\right) = 1\right]\\
&\quad= \Pr[e \overset{R}{\leftarrow} [\![(\{E_1\}_K, K)]\!]_{\Pi(\eta)} : B(1^\eta, e) = 1]\\
&\qquad - \Pr[e \overset{R}{\leftarrow} [\![(\{E_2\}_K, K')]\!]_{\Pi(\eta)} : B(1^\eta, e) = 1]\\
&\quad= \Pr[e \overset{R}{\leftarrow} [\![M_1]\!]_{\Pi(\eta)} : B(1^\eta, e) = 1] - \Pr[e \overset{R}{\leftarrow} [\![M_2]\!]_{\Pi(\eta)} : B(1^\eta, e) = 1]\\
&\quad= \mathrm{Adv}^{\mathrm{ind}}_{[\![M_1]\!]_{\Pi(\eta)},[\![M_2]\!]_{\Pi(\eta)}}(B),
\end{aligned}$$

where the second equality follows from the definition of A, and the third follows from the definition of $\mathsf{Sample}_{\Pi[\eta]}$. Then A is a weak key-authenticity test for E_1, E_2, as required. This completes the necessity part of the proof.

2.3.2 Admittance of WKA-EXP Tests is Sufficient for Formal Encryption to be Computationally-Complete

Here, we prove the *if* part of Theorem 2.3.2. For $j \in \{1, 2\}$, let E_j be an acyclic expression, T_{E_j} its derivation tree over V_{E_j}, $r_{E_j} \in V_{E_j}$ its root, $R_{E_j} \subseteq V_{E_j}$ its set of reachable nodes, $T_{E_j}^{R_{E_j}}$ the tree of reachable nodes. Let $\eta \in \mathbb{N}$ be a security parameter for Π and let e be a sample from either $[\![E_1]\!]_{\Pi(\eta)}$ or $[\![E_2]\!]_{\Pi(\eta)}$. We assume that η is large enough so that $e \in \text{Plain}_{\Pi[\eta]}$ (cf. Section 2.2.2.1). Let $S = V_{E_1} \times V_{E_2} \times \{0,1\}^*$ and let $O_{E_1,E_2,e} : 2^S \to 2^S$, $\text{TEST} : 2^S \to \{\text{true}, \text{false}\}$ be defined as in Figure 2.1. The *powers* of $O_{E_1,E_2,e}$ are defined as follows:

$$O^0_{E_1,E_2,e} = \emptyset$$

$$O^i_{E_1,E_2,e} = O_{E_1,E_2,e}(O^{i-1}_{E_1,E_2,e}) \qquad \text{for all } i \in \mathbb{N}^+$$

We note the following simple properties of $O_{E_1,E_2,e}$ and TEST. It is easy to verify that $O_{E_1,E_2,e}$ is monotonic. Also, observe that for all i, $O^i_{E_1,E_2,e} \subseteq O^{i+1}_{E_1,E_2,e}$, by induction: the base case holds trivially; as for the step, $O^i_{E_1,E_2,e} = O_{E_1,E_2,e}(O^{i-1}_{E_1,E_2,e}) \subseteq O_{E_1,E_2,e}(O^i_{E_1,E_2,e}) = O^{i+1}_{E_1,E_2,e}$, by the hypothesis and the monotonicity of $O_{E_1,E_2,e}$. Finally, note that for all $A, B \in 2^S$ such that $A \subseteq B$, if $\text{TEST}(B)$ holds then $\text{TEST}(A)$ holds; it is also true that if $\text{TEST}(A)$ does not hold, then $\text{TEST}(B)$ does not hold.

We prove the contrapositive: we assume that $E_1 \not\cong E_2$, and show that $[\![E_1]\!]_{\Pi(\eta)} \overset{c}{\not\approx}$

$[\![E_2]\!]_{\Pi(\eta)}$.

2.3.2.1 Intuition

We begin by sketching the main ideas behind the proof. Assume $E_1 \not\cong E_2$. To show that $[\![E_1]\!]_{\Pi(\eta)} \overset{c}{\not\approx} [\![E_2]\!]_{\Pi(\eta)}$, we consider an algorithm that simultaneously parses its input e — a sample from either $[\![E_1]\!]_{\Pi(\eta)}$ or $[\![E_2]\!]_{\Pi(\eta)}$ — and expressions E_1, E_2, attempting to construct φ, σ that bear witness to the equivalence of the expressions. By the assumption, this attempt is bound to fail. We show that upon failure, the algorithm has enough parsed information to predict the origin of the sample with a non-negligible probability of success. In some cases, the prediction depends on an application of a weak key-authenticity test for (particular, fixed) expressions to the amassed information.

Specifically, the algorithm computes the powers of the operator $O_{E_1,E_2,e}$, as long as they satisfy the predicate TEST. Let $i \in \mathbb{N}$. Let $V^i_{E_1} = \left\{v_1 \,\middle|\, (v_1, \cdot, \cdot) \in O^i_{E_1,E_2,e}\right\}$, $V^i_{E_2} = \left\{v_2 \,\middle|\, (\cdot, v_2, \cdot) \in O^i_{E_1,E_2,e}\right\}$. Let $j \in \{1, 2\}$. Let $T^{V^i_{E_j}}_{E_j}$ denote the subtree induced by $V^i_{E_j}$ on T_{E_j}. Let O_{E_j} be the operator from the fixpoint characterization of the set of reachable nodes of T_{E_j} (cf. Theorem 2.1.5). We show that as long as TEST$(O^i_{E_1,E_2,e})$ holds, it is the case that:

1. $V^{i+1}_{E_1} = O^{i+1}_{E_1}$ and $V^{i+1}_{E_2} = O^{i+1}_{E_2}$;
2. there exist φ, σ consistent with the requirements of Claim 2.1.1 when restricted to $T^{V^i_{E_1}}_{E_1}$, $T^{V^i_{E_2}}_{E_2}$, $V^i_{E_1}$, and $V^i_{E_2}$ (instead of $T^R_{E_1}$, $T^R_{E_2}$, R_{E_1}, and R_{E_2}).

If TEST does not fail by the $\max(|E_1|, |E_2|)$'s power of $O_{E_1,E_2,e}$, then V_{E_1}, V_{E_2}

$$O_{E_1,E_2,e}(A) =$$

$$\left\{ \begin{array}{c|l} & \text{either:} \\ & \begin{array}{ll} \text{(a)} & u_1 = r_{E_1},\ u_2 = r_{E_2},\ y = e;\ \text{or} \\ \text{(b)} & \exists(v_1, v_2, x) \in A \text{ such that:} \\ & v_1 \text{ is labeled } (M, N) \text{ and has a left child } u_1 \text{ in } T_{E_1}, \\ & v_2 \text{ is labeled } (M', N') \text{ and has a left child } u_2 \text{ in } T_{E_2} \\ & \text{and } x \text{ is of the form } \langle y, z, \text{“pair”}\rangle;\ \text{or} \\ \text{(c)} & \exists(v_1, v_2, x) \in A \text{ such that:} \\ & v_1 \text{ is labeled } (M, N) \text{ and has a right child } u_1 \text{ in } T_{E_1}, \\ (u_1, u_2, y) & v_2 \text{ is labeled } (M', N') \text{ and has a right child } u_2 \text{ in } T_{E_2} \\ \in S & \text{and } x \text{ is of the form } \langle y, z, \text{“pair”}\rangle;\ \text{or} \\ \text{(d)} & \exists(v_1, v_2, x) \in A \text{ and } \exists(w_1, w_2, z) \in A \text{ such that:} \\ & v_1 \text{ is labeled } \{M\}_K \text{ and has a child } u_1 \text{ in } T_{E_1}, \\ & v_2 \text{ is labeled } \{M'\}_{K'} \text{ and has a child } u_2 \text{ in } T_{E_2}, \\ & x \text{ is of the form } \langle c, \text{“ciphertext”}\rangle, \\ & w_1 \text{ is labeled } K, \\ & w_2 \text{ is labeled } K', \\ & z \text{ is of the form } \langle k, \text{“key”}\rangle \\ & \text{and } y = \mathcal{D}_k(c). \end{array} \end{array} \right\}$$

$$\text{TEST}(A) =$$

$$\begin{cases} \textit{true} & \text{if for all } (v_1, v_2, x) \in A, \text{ either:} \\ & \text{(a)}\ \ v_1 \text{ is labeled with } b \in \textbf{Bits} \text{ and } v_2 \text{ is labeled } b;\ \text{or} \\ & \text{(b)}\ \ v_1 \text{ is labeled } K,\ v_2 \text{ is labeled } K' \text{ and for all } (u_1, u_2, y) \in A, \\ & \qquad u_1 \text{ is labeled } K \text{ iff } u_2 \text{ is labeled } K';\ \text{or} \\ & \text{(c)}\ \ v_1 \text{ is labeled } (M, N) \text{ and } v_2 \text{ is labeled } (M', N');\ \text{or} \\ & \text{(d)}\ \ v_1 \text{ is labeled } \{M\}_K,\ v_2 \text{ is labeled } \{M'\}_{K'} \text{ and for all} \\ & \qquad (u_1, u_2, y) \in A,\ u_1 \text{ is labeled } K \text{ iff } u_2 \text{ is labeled } K'. \\ \textit{false} & \text{otherwise.} \end{cases}$$

Figure 2.1: Definitions of $O_{E_1,E_2,e} : 2^S \rightarrow 2^S$, $\text{TEST} : 2^S \rightarrow \{\text{true}, \text{false}\}$.

achieve the sets of reachable nodes of T_{E_1}, T_{E_2}, respectively, by Item 1 above, and so $E_1 \cong E_2$ by Item 2 above, contradicting our assumption. We conclude that TEST must fail on some lower power of $O_{E_1,E_2,e}$; let $i^* \in \mathbb{N}$ be the lowest such power.

We use $O^{i^*}_{E_1,E_2,e}$ to make a prediction, based on the reason TEST fails. Here, we illustrate a case that calls for the use of a weak key-authenticity test for expressions. Assume TEST fails because there exist $(v_1, v_2, x), (u_1, u_2, y) \in O^{i^*}_{E_1,E_2,e}$ such that v_1 is labeled $\{M\}_K$, u_1 is labeled K, v_2 is labeled $\{M'\}_{K'}$, and u_2 is labeled K''. An inductive argument on the powers of our operator shows that x, y are the sampling labels of either v_1, u_1, respectively, or v_2, u_2, respectively, depending on the origin of e. Let $x = \langle c, \text{“ciphertext”}\rangle, y = \langle k, \text{“key”}\rangle$. In the first case, c is an encryption of a sample from $[\![M]\!]_{\Pi(\eta)}$ with the key k; in the second case, c is an encryption of a sample from $[\![M']\!]_{\Pi(\eta)}$ with some key, and k is a random key. The WKA-EXP-(M,M') test on c and k distinguishes these cases with a non-negligible probability of success.

We conclude the proof by noting that the above procedure is efficient.

2.3.2.2 The Parsing Lemma

We begin by showing that the powers of $O_{E_1,E_2,e}$ effectively parse e in a manner consistent with the structures of E_1 or E_2.

Lemma 2.3.3 (Parsing lemma). *Fix $j \in \{1, 2\}$. If $e \xleftarrow{R} [\![E_j]\!]_{\Pi(\eta)}$, then for all i and for all $(u_1, u_2, y) \in O^i_{E_1,E_2,e}$, y is the sampling label that was assigned to u_j by $\mathsf{Sample}_{\Pi[\eta]}(E_j)$ in the computation of e.*

Proof. Assume $e \overset{R}{\leftarrow} [\![E_j]\!]_{\Pi(\eta)}$. Proceed by induction on i. The base case holds vacuously (as $O^0_{E_1,E_2,e} = \emptyset$). As for the step, assume $(u_1, u_2, y) \in O^{i+1}_{E_1,E_2,e}$; then either:

- $u_1 = r_{E_1}$, $u_2 = r_{E_2}$, $y = e$, as in part (a) of the definition of $O_{E_1,E_2,e}$. By the definition of $\mathsf{Sample}_{\Pi[\eta]}$ on E_j, e is indeed the sampling label assigned to u_j while computing e.

- $\exists(v_1, v_2, x) \in O^i_{E_1,E_2,e}$, v_j is labeled (M, N), u_j is its left child in T_{E_j} and x is of the form $\langle y, z, \text{“pair”}\rangle$, as in part (b) of the definition of $O_{E_1,E_2,e}$. By the induction hypothesis, x is the sampling label given to v_j by $\mathsf{Sample}_{\Pi[\eta]}(E_j)$ while computing e. By the definition of $\mathsf{Sample}_{\Pi[\eta]}$ on E_j, y is indeed the sampling label given to u_j while computing e. The symmetric case (as in part (c) of the definition of $O_{E_1,E_2,e}$) is similar.

- $\exists(v_1, v_2, x), (w_1, w_2, z) \in O^i_{E_1,E_2,e}$ such that v_j is labeled $\{M\}_K$, u_j is its child in T_{E_j}, x is of the form $\langle c, \text{“ciphertext”}\rangle$, w_j is labeled K, z is of the form $\langle k, \text{“key”}\rangle$ and $y = \mathcal{D}_k(c)$, as in part (d) of the definition of $O_{E_1,E_2,e}$. By the induction hypothesis, x and z are the sampling labels assigned to v_j, w_j by $\mathsf{Sample}_{\Pi[\eta]}(E_j)$, respectively, while computing e. As the sampling algorithm consistently assigns computational keys across a derivation tree, k must have been the key used to encrypt the sampling label of u_j when computing c. By the definition of $\mathsf{Sample}_{\Pi[\eta]}$ on E_j, y then is indeed the sampling label given to u_j while computing e.

$\boxtimes$

2.3.2.3 The Isomorphism Lemma

As we proceed, we would like to see what TEST "tells us" when applied to a power of $O_{E_1,E_2,e}$. The following notation, referring to various restrictions of a power of $O_{E_1,E_2,e}$, is instrumental. For all $i \in \mathbb{N}$ let:

$$V^i_{E_1} = \left\{ v_1 \,\middle|\, (v_1, \cdot, \cdot) \in O^i_{E_1,E_2,e} \right\}$$

$$V^i_{E_2} = \left\{ v_2 \,\middle|\, (\cdot, v_2, \cdot) \in O^i_{E_1,E_2,e} \right\}$$

$$\varphi^i = \left\{ (v_1, v_2) \,\middle|\, (v_1, v_2, \cdot) \in O^i_{E_1,E_2,e} \right\}$$

$$\text{Keys}^i_{E_1} = \left\{ K \,\middle|\, \exists (v_1, \cdot, \cdot) \in O^i_{E_1,E_2,e} \text{ such that } v_1 \text{ is labeled } K \right\}$$

$$\text{Keys}^i_{E_2} = \left\{ K \,\middle|\, \exists (\cdot, v_2, \cdot) \in O^i_{E_1,E_2,e} \text{ such that } v_2 \text{ is labeled } K \right\}$$

$$\sigma^i = \left\{ (K_1, K_2) \,\middle|\, \begin{array}{l} \exists (v_1, v_2, \cdot) \in O^i_{E_1,E_2,e} \text{ such that } v_1 \text{ is} \\ \text{labeled } K_1 \text{ and } v_2 \text{ is labeled } K_2 \end{array} \right\}$$

In addition, for $j \in \{1, 2\}$, $A \subseteq V_{E_j}$, let $T^A_{E_j}$ denote the subgraph induced by A on T_{E_j}. We claim the following:

Lemma 2.3.4 (Isomorphism lemma)**.** *For all i, $\varphi^i : V^i_{E_1} \to V^i_{E_2}$ is an isomorphism of $T^{V^i_{E_1}}_{E_1}$ and $T^{V^i_{E_2}}_{E_2}$ as rooted, ordered trees. Furthermore, if* $\text{TEST}(O^i_{E_1,E_2,e})$ *holds, then $\sigma^i : \text{Keys}^i_{E_1} \to \text{Keys}^i_{E_2}$ is a bijection, and for all $v \in V^i_{E_1}$,*

1. *if v is labeled with a bit, then $\varphi^i(v)$ is labeled with an identical bit.*

2. *if v is labeled K, then $\varphi^i(v)$ is labeled with $\sigma^i(K)$.*

3. *if v is labeled (M, N), then $\varphi^i(v)$ is labeled (M', N').*

4. *if v is labeled $\{M\}_K$ and there exists a $u \in V^i_{E_1}$ labeled K, then $\varphi^i(v)$ is labeled $\{M'\}_{\sigma^i(K)}$ and $\varphi^i(u)$ is labeled $\sigma^i(K)$.*

5. *if v is labeled $\{M\}_K$ and there does not exists a $u \in V^i_{E_1}$ labeled K, then $\varphi^i(v)$ is labeled $\{M'\}_{K'}$ and there does not exist a $u' \in V^i_{E_2}$ labeled K'.*

Note the similarity to Claim 2.1.1.

Proof. We prove that for all i, $\varphi^i : V^i_{E_1} \to V^i_{E_2}$ is an isomorphism of $T^{V^i_{E_1}}_{E_1}$ and $T^{V^i_{E_2}}_{E_2}$ as rooted, ordered trees; the rest of the claim follows in a straightforward manner from the definition of TEST.

First, we show that φ^i is a function, by induction. The base case holds vacuously. As for the step, assume $(u_1, u_2), (u_1, u'_2) \in \varphi^i$. Then there exist $(u_1, u_2, y), (u_1, u'_2, y') \in O^i_{E_1,E_2,e}$. u_1 is either the root of T_{E_1} or has a unique parent v in T_{E_1}. We have that either:

- $u_1 = r_{E_1}$, $u_2 = u'_2 = r_{E_2}$ (and $y = y' = e$) as in part (a) of the definition of $O_{E_1,E_2,e}$.

- There exists a $(v_1, v_2, x) \in O^{i-1}_{E_1,E_2,e}$, related to (u_1, u_2, y) as in part (b) of the definition of $O_{E_1,E_2,e}$, and there exists a $(v_1, v'_2, x') \in O^{i-1}_{E_1,E_2,e}$, related to (u_1, u'_2, y') as in part (b) of the definition of $O_{E_1,E_2,e}$. In particular, v_1 has u_1 as a left child in T_{E_1}, and v_2, v'_2 have u_1, u'_2 as left children, respectively, in T_{E_2}. By the induction hypothesis, φ^{i-1} is a function and so $v_2 = v'_2$. It follows that $u_2 = u'_2$. The symmetric case (related to part (c) of the definition of $O_{E_1,E_2,e}$) is similar.

- There exist $(v_1, v_2, x), (w_1, w_2, z) \in O^{i-1}_{E_1,E_2,e}$, related to (u_1, u_2, y) as in part (d) of the definition of $O_{E_1,E_2,e}$, and there exist $(v_1, v'_2, x'), (w'_1, w'_2, z') \in O^{i-1}_{E_1,E_2,e}$

related to (u_1, u_2', y') as in part (d) of the definition of $O_{E_1,E_2,e}$. In particular, v_1 has u_1 as a child in T_{E_1}, and v_2, v_2' have u_2, u_2' as their respective children in T_{E_2}. By the induction hypothesis, φ^{i-1} is a function and so $v_2 = v_2'$. It follows that $u_2 = u_2'$.

The argument for φ^i being one-to-one is completely symmetric.

All that is left to be shown is that v has children $(u_1, u_2, \ldots, u_k)$ in $T_{E_1}^{V_{E_1}^i}$ iff $\varphi^i(v)$ has children $(\varphi^i(u_1), \varphi^i(u_2), \ldots, \varphi^i(u_k))$ in $T_{E_2}^{V_{E_2}^i}$. For the *only-if* part, assume that both v_1, labeled (M, N), and its left child u_1 are in $T_{E_1}^{V_{E_1}^i}$. By definition, $v_1, u_1 \in V_{E_1}^i$ and so there exist $(v_1, v_2, x), (u_1, u_2, y) \in O_{E_1,E_2,e}^i$. By definition, $\varphi^i(v_1) = v_2$, $\varphi^i(u_1) = u_2$, $v_2, u_2 \in V_{E_2}^i$ and are also in $T_{E_2}^{V_{E_2}^i}$; we have to show that u_2 is the left child of v_2. As $(u_1, u_2, y) \in O_{E_1,E_2,e}^i$, there exists a $(v_1, v_2', x') \in O_{E_1,E_2,e}^{i-1}$ related to it as in part (b) of the definition of $O_{E_1,E_2,e}$. In particular, u_2 is the left child of v_2'. Now $O_{E_1,E_2,e}^{i-1} \subseteq O_{E_1,E_2,e}^i$, therefore $(v_1, v_2', x') \in O_{E_1,E_2,e}^i$. φ^i is a function, and so it must be the case that $v_2 = v_2'$. u_2 is thus shown to be the left child of v_2. Similar arguments establish the *only-if* part for a node labeled (M, N) and its right child; and for a node labeled $\{M\}_K$ and its child. The *if* part is symmetric. $\boxtimes$

2.3.2.4 The Reachable-Sets Lemma

We now relate the powers of $O_{E_1,E_2,e}$ to the powers of the operators O_{E_1}, O_{E_2}, whose fixpoints correspond to the sets of reachable nodes of T_{E_1}, T_{E_2}, respectively. For $j \in \{1, 2\}$, let $O_{E_j} : 2^{V_{E_j}} \rightarrow 2^{V_{E_j}}$ be the operator from the fixpoint characterizations of the set of reachable nodes of T_{E_j} (cf. Theorem 2.1.5).

Lemma 2.3.5 (Reachable-Sets Lemma). *For all $i \in \mathbb{N}$, if* $\mathrm{TEST}(O^i_{E_1,E_2,e})$ *holds, then $V^{i+1}_{E_1} = O^{i+1}_{E_1}$ and $V^{i+1}_{E_2} = O^{i+1}_{E_2}$.*

Proof. We prove that for all $i \in \mathbb{N}$, if $\mathrm{TEST}(O^i_{E_1,E_2,e})$ holds, then $V^{i+1}_{E_1} = O^{i+1}_{E_1}$; the proof for $V^{i+1}_{E_2} = O^{i+1}_{E_2}$ is similar.

We proceed with induction on i. For the base case, note that $\mathrm{TEST}(O^0_{E_1,E_2,e})$ holds vacuously and that $V^1_{E_1} = O^1_{E_1} = r_{E_1}$. As for the step, assume that the claim holds for $(i-1)$, and assume that $\mathrm{TEST}(O^i_{E_1,E_2,e})$ holds. Note that $O^{i-1}_{E_1,E_2,e} \subseteq O^i_{E_1,E_2,e}$, and so $\mathrm{TEST}(O^{i-1}_{E_1,E_2,e})$ holds as well.

To show that $V^{i+1}_{E_1} \subseteq O^{i+1}_{E_1}$, let $u_1 \in V^{i+1}_{E_1}$. Then there exists a $(u_1, u_2, y) \in O^{i+1}_{E_1,E_2,e}$. It is the case that either:

- $u_1 = r_{E_1}$, $u_2 = r_{E_2}$, $y = e$ as in part (a) of the definition of $O_{E_1,E_2,e}$. But r_{E_1} is also an element of $O^{i+1}_{E_1}$, by the definition of O_{E_1}.

- $\exists (v_1, v_2, x) \in O^i_{E_1,E_2,e}$, related to (u_1, u_2, y) as in part (b) of the definition of $O_{E_1,E_2,e}$. In particular, v_1 is labeled (M, N) and has u_1 as a left child in T_{E_1}. Now $v_1 \in V^i_{E_1} = O^i_{E_1}$ by the induction hypothesis, and so $u_1 \in O^{i+1}_{E_1}$ by the definition of O_{E_1}. The symmetric case (as in part (c) of the definition of $O_{E_1,E_2,e}$) is similar.

- $\exists (v_1, v_2, x), (w_1, w_2, z) \in O^i_{E_1,E_2,e}$, related to (u_1, u_2, y) as in part (d) of the definition of $O_{E_1,E_2,e}$. In particular, v_1 is labeled $\{M\}_K$ and has u_1 as a child in T_{E_1}, w_1 is labeled K. Now $v_1, w_1 \in V^i_{E_1} = O^i_{E_1}$ by the induction hypothesis, and so $u_1 \in O^{i+1}_{E_1}$ by the definition of O_{E_1}.

We now show that $O^{i+1}_{E_1} \subseteq V^{i+1}_{E_1}$. Let $u_1 \in O^{i+1}_{E_1}$. Then either:

- $u_1 = r_{E_1}$, as in part (a) of the definition of O_{E_1}. By part (a) of the definition of $O_{E_1,E_2,e}$, $(r_{E_1}, r_{E_2}, e) \in O^{i+1}_{E_1,E_2,e}$, and so $u_1 = r_{E_1} \in V^{i+1}_{E_1}$.

- $\exists v_1 \in O^i_{E_1}$ such that v_1 is labeled (M, N) and has u_1 as a left child in T_{E_1}, as in part (b) of the definition of O_{E_1}. By the induction hypothesis, $v_1 \in V^i_{E_1}$. Then $\exists (v_1, v_2, x) \in O^i_{E_1,E_2,e}$. As TEST$(O^i_{E_1,E_2,e})$ holds, it must be the case that v_2 is labeled (M', N') and so has a left child u_2 in T_{E_2}. Now by the Parsing Lemma, x is the sampling label of either v_1 or v_2, depending on the origin of e. At any rate, x is of the form $\langle y, z, \text{“pair”}\rangle$. By part (b) of the definition of $O_{E_1,E_2,e}$, $(u_1, u_2, y) \in O^{i+1}_{E_1,E_2,e}$ (where u_2 is the left child of v_2 in T_{E_2}). It follows that $u_1 \in V^{i+1}_{E_1}$. The symmetric case (related to part (c) of the definition of O_{E_1}) is similar.

- $\exists v_1, w_1 \in O^i_{E_1}$ such that v_1 is labeled $\{M\}_K$ and has u_1 as a left child in T_{E_1}, w_1 is labeled K, as in part (d) of the definition of O_{E_1}. By the induction hypothesis, $v_1, w_1 \in V^i_{E_1}$, and so $\exists (v_1, v_2, x), (w_1, w_2, z) \in O^i_{E_1,E_2,e}$. As TEST$(O^i_{E_1,E_2,e})$ holds, it must be the case that v_2 is labeled $\{M'\}_{K'}$ and w_2 is labeled K'. By the Parsing Lemma, x, z are the sampling labels, respectively, of either v_1, w_1 or v_2, w_2, depending on the origin of e. In both cases, x and z must be of the forms $\langle c, \text{“ciphertext”}\rangle$ and $\langle k, \text{“key”}\rangle$, respectively, by the definition of the sampling procedure. By part (d) of the definition of $O_{E_1,E_2,e}$, $(u_1, u_2, \mathcal{D}_k(c)) \in O^{i+1}_{E_1,E_2,e}$ (where u_2 is the child of v_2 in T_{E_2}). It follows that $u_1 \in V^{i+1}_{E_1}$.

We conclude that $V^{i+1}_{E_1} = O^{i+1}_{E_1}$. ⊠

2.3.2.5 Constructing a Distinguisher

We now turn to the main part of our argument. Informally, we use the Reachable-Sets Lemma and the fixpoint characterization of the set of reachable nodes to show that if $\mathrm{TEST}(O^{i'}_{E_1,E_2,e})$ holds for a large enough i' (linear in the sizes of E_1, E_2), then $V^{i'}_{E_1}, V^{i'}_{E_2}$ achieve the sets of reachable nodes of T_{E_1}, T_{E_2}, respectively. At that point, the Isomorphism Lemma guarantees that $\varphi^{i'}, \sigma^{i'}$ witness the equivalence of E_1, E_2, in contradiction to our assumption that $E_1 \not\cong E_2$. Therefore, there must exist an $i^* \leq i'$ for which $\mathrm{TEST}(O^{i^*}_{E_1,E_2,e})$ fails. We use $O^{i^*}_{E_1,E_2,e}$ and the Parsing Lemma to construct a distinguisher of $[\![E_1]\!]_{\Pi(\eta)}, [\![E_2]\!]_{\Pi(\eta)}$.

Formally, recall our assumption that $E_1 \not\cong E_2$. Towards a contradiction, further assume that for all $i \in \mathbb{N}$, $\mathrm{TEST}(O^{i}_{E_1,E_2,e})$ holds. Then for all $i \in \mathbb{N}$, $V^i_{E_1} = O^i_{E_1}$ and $V^i_{E_2} = O^i_{E_2}$ by the Reachable-Sets Lemma (and a straightforward verification of the case where $i = 0$). Let $|E_1| = n_1$ and let $i_1 \leq n_1$ be such that for all $j \geq i_1$, $O^j_{E_1} = R_{E_1}$, as guaranteed by the fixpoint characterization of R_{E_1} (cf. Theorem 2.1.5); similarly, for $|E_2| = n_2$, let $i_2 \leq n_2$ be such that for all $j \geq i_2$, $O^j_{E_2} = R_{E_2}$. Without loss of generality, assume $i_1 \leq i_2$. Then for all $j \geq i_1$, $V^j_{E_1} = O^j_{E_1} = R_{E_1}$ (i.e., $V^j_{E_1}$ fixes). But for all such j, $V^j_{E_1}$ is bijected onto $V^j_{E_2}$ via φ^j by the Isomorphism Lemma; it is also the case that φ^j is an extension of φ^{i_1} (as $O^i_{E_1,E_2,e} \subseteq O^j_{E_1,E_2,e}$ for all $i \leq j$). It follows that for all $j \geq i_1$, $V^j_{E_2}$ is fixed too, and so $V^j_{E_2} = R_{E_2}$ (cf. Lemma 2.1.2 and Theorem 2.1.5). We conclude that for $i_{12} = \min(i_1, i_2)$, we have that $V^{i_{12}}_{E_1} = R_{E_1}$ and $V^{i_{12}}_{E_2} = R_{E_2}$. But then $\varphi^{i_{12}}$ and $\sigma^{i_{12}}$ (when the latter is properly extended to a bijection on **Keys**) witness the equivalence of E_1, E_2 by the Isomorphism Lemma, in contradiction to our assumption.

It follows that if $E_1 \ncong E_2$, there must exist an $i \leq \min(|E_1|, |E_2|)$ such that $\mathrm{TEST}(O^i_{E_1,E_2,e})$ fails. Let i^* be the smallest such i. We argue that given $O^{i^*}_{E_1,E_2,e}$, we have enough information to tell, with non-negligible probability of success, which of $[\![E_1]\!]_{\Pi(\eta)}, [\![E_2]\!]_{\Pi(\eta)}$ e originated from. To that effect, we describe a procedure $\mathsf{Predict}$, that takes η (in unary) and $O^{i^*}_{E_1,E_2,e}$ as inputs, and outputs 1 if it believes $e \overset{R}{\leftarrow} [\![E_1]\!]_{\Pi(\eta)}$, 0 otherwise. The behavior of $\mathsf{Predict}$ depends on the reason for which TEST fails on $O^{i^*}_{E_1,E_2,e}$, as described below. For every possible reason and consequent action, we analyze $\mathsf{Predict}$'s advantage in distinguishing the above-mentioned ensembles; the case discussed in Item 7 (and the symmetric one in Item 8) is where our "thunder" lies — this is where $\mathsf{Predict}$ uses a weak key-authenticity test for two specific expressions, admitted by Π.

The possible reasons for TEST to fail on $O^{i^*}_{E_1,E_2,e}$, and the corresponding actions by $\mathsf{Predict}$ (with their chances of success) are the following:

1. $\exists (v_1, v_2, x) \in O^{i^*}_{E_1,E_2,e}$ such that v_1 is labeled with a bit but v_2 is not. If $e \overset{R}{\leftarrow} [\![E_1]\!]_{\Pi(\eta)}$, then x is the sampling label of v_1 by the Parsing Lemma, and is of the form $\langle b, \text{"bit"}\rangle$ by the labeling of v_1 and the definition of Sample. If $e \overset{R}{\leftarrow} [\![E_2]\!]_{\Pi(\eta)}$, then x is the sampling label of v_2 and is *not* of the form $\langle b', \text{"bit"}\rangle$. Let $\mathsf{Predict}_{\text{case-1}}$ be 1 if $x = \langle b, \text{"bit"}\rangle$, 0 otherwise. Clearly, $\mathrm{Adv}^{\mathrm{ind}}_{[\![E_1]\!]_{\Pi(\eta)},[\![E_2]\!]_{\Pi(\eta)}}(\mathsf{Predict}_{\text{case-1}}) = 1$.

2. $\exists (v_1, v_2, x) \in O^{i^*}_{E_1,E_2,e}$ such that v_1 is labeled with a bit b and v_2 with $\bar{b}$ (i.e., negated b). As in case (1), we let $\mathsf{Predict}_{\text{case-2}}$ be 1 if $x = \langle b, \text{"bit"}\rangle$, 0 otherwise.

Here too we have $\mathrm{Adv}^{\mathrm{ind}}_{[\![E_1]\!]_{\Pi(\eta)},[\![E_2]\!]_{\Pi(\eta)}}(\mathsf{Predict}_{\text{case-2}}) = 1$.

3. $\exists(v_1, v_2, x) \in O^{i^*}_{E_1,E_2,e}$ such that v_1 is labeled with a key but v_2 is not. As in case (1), we let $\mathsf{Predict}_{\text{case-3}}$ be 1 if $x = \langle k, \text{“key”}\rangle$, 0 otherwise. Once again, it is clear that $\mathrm{Adv}^{\mathrm{ind}}_{[\![E_1]\!]_{\Pi(\eta)},[\![E_2]\!]_{\Pi(\eta)}}(\mathsf{Predict}_{\text{case-3}}) = 1$.

4. $\exists(v_1, v_2, x), (u_1, u_2, y) \in O^{i^*}_{E_1,E_2,e}$ such that v_1 and u_1 are labeled K, v_2 is labeled K' and u_2 is labeled K'' (where $K' \neq K''$). If $e \overset{R}{\leftarrow} [\![E_1]\!]_{\Pi(\eta)}$, then x and y are the sampling labels of v_1 and u_1, respectively, by the Parsing Lemma. Since both are labeled with the same key, Sample computes $k \overset{R}{\leftarrow} \mathcal{K}(1^\eta)$ and assigns to both nodes a sampling label $\langle k, \text{“key”}\rangle$. If $e \overset{R}{\leftarrow} [\![E_2]\!]_{\Pi(\eta)}$, then x and y are the sampling labels of v_2 and u_2, respectively. But since they are labeled with distinct formal keys, Sample computes $k, k' \overset{R}{\leftarrow} \mathcal{K}(1^\eta)$ and assigns to one a sampling label $\langle k, \text{“key”}\rangle$ and to the other a sampling label $\langle k', \text{“key”}\rangle$. Let $\mathsf{Predict}_{\text{case-4}}$ be 1 if $x = y = \langle k, \text{“key”}\rangle$, 0 otherwise. We have:

$$\begin{aligned}
&\mathrm{Adv}^{\mathrm{ind}}_{[\![E_1]\!]_{\Pi(\eta)},[\![E_2]\!]_{\Pi(\eta)}}(\mathsf{Predict}_{\text{case-4}})\\
&\quad= \Pr[e \overset{R}{\leftarrow} [\![E_1]\!]_{\Pi(\eta)}; B \leftarrow O^{i^*}_{E_1,E_2,e} : \mathsf{Predict}_{\text{case-4}}(1^\eta, B) = 1]\\
&\qquad - \Pr[e \overset{R}{\leftarrow} [\![E_2]\!]_{\Pi(\eta)}; B \leftarrow O^{i^*}_{E_1,E_2,e} : \mathsf{Predict}_{\text{case-4}}(1^\eta, B) = 1]\\
&\quad= \Pr[k \overset{R}{\leftarrow} \mathcal{K}(1^\eta) : k = k] - \Pr[k, k' \overset{R}{\leftarrow} \mathcal{K}(1^\eta) : k = k']\\
&\quad= 1 - \Pr[k, k' \overset{R}{\leftarrow} \mathcal{K}(1^\eta) : k = k']\\
&\quad= \Pr[k, k' \overset{R}{\leftarrow} \mathcal{K}(1^\eta) : k \neq k'],
\end{aligned}$$

which is non-negligible in η as $\mathcal{K}$ distributes keys decently.

5. $\exists(v_1, v_2, x), (u_1, u_2, y) \in O^{i^*}_{E_1,E_2,e}$ such that v_2 and u_2 are labeled K, v_1 is labeled K' and u_1 is labeled K'' (where $K' \neq K''$). As in case (4), we let $\mathsf{Predict}_{\text{case-5}}$

be 1 if $x = \langle k, \text{“key”}\rangle$ and $y = \langle k', \text{“key”}\rangle$ where $k \neq k'$, 0 otherwise. As before, we have that $\mathrm{Adv}^{\mathrm{ind}}_{[\![E_1]\!]_{\Pi(\eta)},[\![E_2]\!]_{\Pi(\eta)}}(\mathsf{Predict}_{\text{case-5}})$ is non-negligible (as a function of η).

6. $\exists(v_1, v_2, x) \in O^{i*}_{E_1,E_2,e}$ such that v_1 is labeled (M, N) but v_2 is not labeled (M', N'). As in case (1), we let $\mathsf{Predict}_{\text{case-6}}$ be 1 if $x = \langle m, n, \text{“pair”}\rangle$, 0 otherwise. Clearly, we have that $\mathrm{Adv}^{\mathrm{ind}}_{[\![E_1]\!]_{\Pi(\eta)},[\![E_2]\!]_{\Pi(\eta)}}(\mathsf{Predict}_{\text{case-6}}) = 1$.

7. $\exists(v_1, v_2, x), (u_1, u_2, y) \in O^{i*}_{E_1,E_2,e}$ such that v_1 is labeled $\{M\}_K$, v_2 is labeled $\{M'\}_{K'}$, u_1 is labeled K and u_2 is labeled K''. If $e \overset{R}{\leftarrow} [\![E_1]\!]_{\Pi(\eta)}$, then x and y are the sampling labels of v_1 and u_1, respectively, by the Parsing Lemma. Because of their labeling and the consistent assignment of keys by Sample across a derivation tree, x must be of the form $\langle c, \text{“ciphertext”}\rangle$, y of the form $\langle k, \text{“key”}\rangle$ where $k \overset{R}{\leftarrow} \mathcal{K}(1^\eta)$, and c is the encryption of some string with k. If $e \overset{R}{\leftarrow} [\![E_2]\!]_{\Pi(\eta)}$, then x and y are the sampling labels of v_2 and u_2, respectively. Because of their labeling and the definition of Sample, x must be of the form $\langle c, \text{“ciphertext”}\rangle$, y of the form $\langle k, \text{“key”}\rangle$ where $k \overset{R}{\leftarrow} \mathcal{K}(1^\eta)$, and c is the encryption of some string with k' where $k' \overset{R}{\leftarrow} \mathcal{K}(1^\eta)$.

   Let A be the WKA-EXP-(M, M') test admitted by Π. We let $\mathsf{Predict}_{\text{case-7}}$ be

1 if $A(1^\eta, c, k) = 1$, 0 otherwise. We have:

$$\begin{aligned}
&\mathrm{Adv}^{\mathrm{ind}}_{[\![E_1]\!]_{\Pi(\eta)},[\![E_2]\!]_{\Pi(\eta)}}(\mathsf{Predict}_{\text{case-7}}) \\
&\quad = \Pr[e \xleftarrow{R} [\![E_1]\!]_{\Pi(\eta)}; B \leftarrow O^{i*}_{E_1,E_2,e} : \mathsf{Predict}_{\text{case-7}}(1^\eta, B) = 1] \\
&\qquad - \Pr[e \xleftarrow{R} [\![E_2]\!]_{\Pi(\eta)}; B \leftarrow O^{i*}_{E_1,E_2,e} : \mathsf{Predict}_{\text{case-7}}(1^\eta, B) = 1] \\
&\quad = \Pr[e \xleftarrow{R} [\![M]\!]_{\Pi(\eta)}; k \xleftarrow{R} \mathcal{K}(1^\eta); c \xleftarrow{R} \mathcal{E}_k(e) : A(1^\eta, c, k) = 1] \\
&\qquad - \Pr[e \xleftarrow{R} [\![M']\!]_{\Pi(\eta)}; k, k' \xleftarrow{R} \mathcal{K}(1^\eta); c \xleftarrow{R} \mathcal{E}_{k'}(e) : A(1^\eta, c, k) = 1] \\
&\quad = \mathrm{Adv}^{\text{wka-exp}}_{\Pi[\eta],M,M'}(A),
\end{aligned}$$

which is non-negligible (as a function of η) by the definition of the WKA-EXP test for expressions.

8. $\exists (v_1, v_2, x), (u_1, u_2, y) \in O^{i*}_{E_1,E_2,e}$ such that v_1 is labeled $\{M\}_{K'}$, v_2 is labeled $\{M'\}_K$, u_1 is labeled K'' and u_2 is labeled K. As in case (7), we have $x = \langle c, \text{“ciphertext”}\rangle$ and $y = \langle k, \text{“key”}\rangle$. Let A be the WKA-EXP-(M, M') test admitted by Π, and let $\mathsf{Predict}_{\text{case-8}}$ be 1 if $A(1^\eta, c, k) = 0$, 0 otherwise. As in case (7), we get that $\mathrm{Adv}^{\mathrm{ind}}_{[\![E_1]\!]_{\Pi(\eta)},[\![E_2]\!]_{\Pi(\eta)}}(\mathsf{Predict}_{\text{case-8}})$ is non-negligible (as a function of η).

Having covered all possible cases, we conclude that $\mathrm{Adv}^{\mathrm{ind}}_{[\![E_1]\!]_{\Pi(\eta)},[\![E_2]\!]_{\Pi(\eta)}}(\mathsf{Predict})$ is non-negligible (as a function of η).

Putting it together, a distinguisher D_{E_1,E_2} for $[\![E_1]\!]_{\Pi(\eta)}, [\![E_1]\!]_{\Pi(\eta)}$ will act as follows:

```
$D_{E_1,E_2}(1^\eta, e)$
  $B \leftarrow \emptyset$;
```

```
while (TEST($B$) holds)
    $B \leftarrow O_{E_1,E_2,e}(B)$;
Output $\mathsf{Predict}(1^\eta, B)$.
```

Note that:

$$\begin{aligned}
&\mathrm{Adv}^{\mathrm{ind}}_{[\![E_1]\!]_{\Pi(\eta)},[\![E_2]\!]_{\Pi(\eta)}}(D_{E_1,E_2}) \\
&\quad = \Pr[e \xleftarrow{R} [\![E_1]\!]_{\Pi(\eta)} : D_{E_1,E_2}(1^\eta, e) = 1] \\
&\qquad - \Pr[e \xleftarrow{R} [\![E_2]\!]_{\Pi(\eta)} : D_{E_1,E_2}(1^\eta, e) = 1] \\
&\quad = \Pr[e \xleftarrow{R} [\![E_1]\!]_{\Pi(\eta)}; B \leftarrow O^{i^*}_{E_1,E_2,e} : \mathsf{Predict}(1^\eta, B) = 1] \\
&\qquad - \Pr[e \xleftarrow{R} [\![E_2]\!]_{\Pi(\eta)}; B \leftarrow O^{i^*}_{E_1,E_2,e} : \mathsf{Predict}(1^\eta, B) = 1] \\
&\quad = \mathrm{Adv}^{\mathrm{ind}}_{[\![E_1]\!]_{\Pi(\eta)},[\![E_2]\!]_{\Pi(\eta)}}(\mathsf{Predict}),
\end{aligned}$$

which is non-negligible (as a function of η). Finally, it is simple to verify that D runs in time polynomial in η. This completes the proof of the sufficiency part of our main theorem.

2.4 How Weak Key-Authenticity Relates to Other Cryptographic Notions

Here, we relate the notion of weak key-authenticity with other relevant cryptographic notions. We begin by strengthening the notion of the admittance of weak key-authenticity tests for expressions; the strengthened flavor considers the admittance of a *single*, all-purpose test, hereby referred to as the *weak key-authenticity test*, that distinguishes any ciphertext and the key it was encrypted with from any ciphertext and a random key, with a non-negligible probability. We stress that the strengthened test is defined in terms that are *independent* of the formal language of the preceding sections. We compare the strengthened flavor with the notions of confusion-freedom and authenticated encryption, previously discussed in the literature in the context of a computational-completeness result for formal symmetric-encryption [AJ01, MW04a]. Specifically, we show that the requirement that an encryption scheme admits a weak key-authenticity test is *strictly weaker* than the requirement that it be confusion-free, as defined there (which, in turn, is enough to show it is strictly weaker than authenticated encryption as well). To that effect, we present an encryption scheme that admits a weak key-authenticity test but *is not* confusion-free. The scheme we present is also Type-0. It therefore satisfies the soundness criterion of [AR02], our completeness criterion, but *not* the previous completeness criterion of [MW04a]. The notions we present and the methods used to achieve the admittance of a weak key-authenticity test should be of independent interest.

Informally, confusion-freedom captures the ability of a decryption algorithm to distinguish a ciphertext and the key it was encrypted with from a ciphertext and a random key with almost full certainty. In contrast, the weak key-authenticity test is required to distinguish the two with merely a non-negligible probability. We will separate the notions in a strong sense, pertaining directly to the gap in their required distinguishing certainties (as opposed to pertaining to the placement of the distinguisher—inside or outside the decryption algorithm).

We begin with formal definitions of the notions at hand. Confusion-freedom is defined as it appears in the completeness result of [MW04a]; our proofs can be modified to accommodate the version of [AJ01] as well.

Definition 2.4.1 (Confusion-Freedom)**.** Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an encryption scheme, $\eta \in \mathbb{N}$ a security parameter, and $D[\eta] = \{D_1[\eta], \ldots, D_l[\eta]\}$ a series of finite sets of distributions. For $1 \leq i \leq l$, define:

$$\mathrm{Adv}^{\mathrm{cf}}_{\Pi[\eta],D[\eta],i} = \Pr[k, k' \overset{R}{\leftarrow} \mathcal{K}(1^\eta); x \overset{R}{\leftarrow} D_i[\eta] : \mathcal{D}_{k'}(\mathcal{E}_k(x)) \neq \bot].$$

We say that Π is *confusion-free* (*CF* for short) if for any $1 \leq i \leq l$, $\mathrm{Adv}^{\mathrm{cf}}_{\Pi[\eta],D[\eta],i}$ is negligible (as a function of η). $\diamondsuit$

Next, we define two auxiliary notions that will provide a "middle ground" for comparing WKA-EXP tests with CF.

Definition 2.4.2 (Strong Key-Authenticity Test, Weak Key-Authenticity Test)**.** Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an encryption scheme, $\eta \in \mathbb{N}$ a security parameter. Let $\mathcal{P}_1, \mathcal{P}_2$ (hereby referred to as *plaintext generators*) be probabilistic algorithms that take a security parameter η (provided in unary), and for sufficiently large η always return an $x \in \mathrm{Plain}_{\Pi[\eta]}$; we write $x \overset{R}{\leftarrow} \mathcal{P}_j(1^\eta)$ for $j \in \{1, 2\}$, thinking of x as being drawn from the probability distribution induced by $\mathcal{P}_j(1^\eta)$ on $\{0, 1\}^*$. Let A be an algorithm. Define:

$$\begin{aligned}
&\mathrm{Adv}^{\mathrm{tst}}_{\Pi[\eta],\mathcal{P}_1[\eta],\mathcal{P}_2[\eta]}(A) \\
&\quad = \Pr[x \overset{R}{\leftarrow} \mathcal{P}_1(1^\eta); k \overset{R}{\leftarrow} \mathcal{K}(1^\eta); c \overset{R}{\leftarrow} \mathcal{E}_k(x) : A(1^\eta, c, k) = 1] \\
&\qquad - \Pr[x \overset{R}{\leftarrow} \mathcal{P}_2(1^\eta); k, k' \overset{R}{\leftarrow} \mathcal{K}(1^\eta); c \overset{R}{\leftarrow} \mathcal{E}_k(x) : A(1^\eta, c, k') = 1],
\end{aligned}$$

where tst $\in \{\text{ska}, \text{wka}\}$. We say that Π admits a *strong* (resp., *weak*) *key-authenticity test*, *SKA* (resp., *WKA*) for short, if there exists a probabilistic, polynomial-time algorithm A such that for all probabilistic, polynomial-time algorithms $\mathcal{P}_1, \mathcal{P}_2$, $\text{Adv}^{\text{ska}}_{\Pi[\eta],\mathcal{P}_1[\eta],\mathcal{P}_2[\eta]}(A)$ (resp., $\text{Adv}^{\text{wka}}_{\Pi[\eta],\mathcal{P}_1[\eta],\mathcal{P}_2[\eta]}(A)$) is negligibly close to 1 (resp., is non-negligible) as a function of η. $\diamondsuit$

As for the definition of *integrity of plaintext* security (*INT-PTXT* for short), a flavor of authenticated encryption, we refer the reader to [BN00, KY00] and to [MW04a].

The following diagram depicts relationships between our notions of interest.

$$\begin{tikzcd}
\text{INT-PTXT} \arrow[r] & \text{CF} \arrow[r] & \begin{array}{c}\text{Admittance}\\ \text{of an}\\ \text{SKA test}\end{array} \arrow[r, shift left] & \begin{array}{c}\text{Admittance}\\ \text{of a}\\ \text{WKA test}\end{array} \arrow[l, shift left, "/" marking] \arrow[r] & \begin{array}{c}\text{Admittance}\\ \text{of}\\ \text{WKA-EXP tests}\end{array}
\end{tikzcd}$$

In the above, $A \longrightarrow B$ means that an encryption scheme that meets notion A must also meet notion B; we call such a relationship an *implication*. $A \not\longrightarrow B$ means that an encryption scheme that meets notion A does not necessarily meet notion B; we call such a relationship a *separation*.

The implications in the diagram are mostly straightforward. For INT-PTXT $\longrightarrow$ CF, refer to [MW04a]. For CF $\longrightarrow$ SKA, let $A(1^\eta, c, k)$ be an algorithm that outputs 1 if $\mathcal{D}_k(c) \neq \perp$, 0 otherwise. By noticing that for any $x \in \text{Plain}_{\Pi[\eta]}$, $\mathcal{D}_k(\mathcal{E}_k(x)) \neq \perp$ by the correctness of Π, and by letting $D[\eta]$ be $\{\mathcal{P}_2(1^\eta)\}$, the implication follows. SKA $\longrightarrow$ WKA is trivial. As for WKA $\longrightarrow$ WKA-EXP tests, we let the WKA test serve as a WKA-EXP test for any pair of expressions. As we assumed that for any expression E and for large enough η, $e \xleftarrow{R} [\![E]\!]_{\Pi(\eta)}$ is such that $e \in \text{Plain}_{\Pi[\eta]}$, the implication follows.

The rest of the section is devoted to the separation of WKA from SKA. To that end, we show an encryption scheme that admits a WKA test but does not admit an SKA test. We use a standard construction based on a pseudorandom function family, with an added "weak redundancy". To simplify the exposition, we use a single, constant bit as redundancy; see the end of the section for a generalization.

Let F be a pseudorandom family of functions with a security parameter $\eta \in \mathbb{N}$, key domain $\{0,1\}^\eta$, domain $\{0,1\}^{l(\eta)}$ and range $\{0,1\}^{L(\eta)}$ (where l, L are polynomials in η); let ϵ be a negligible function such that $\text{Adv}^{\text{prf}}_{F[\eta]}(A) \le \epsilon(\eta)$ for any probabilistic, polynomial-time algorithm A. We use $x_1 x_2 \cdots x_m$ to denote the individual bits of a string $x \in \{0,1\}^m$. We use $\circ$ to denote the concatenation operator on strings of bits, $\oplus$ to denote the bitwise XOR operator on strings of bits of equal length.

Define an encryption scheme $\Pi^* = (\mathcal{K}^*, \mathcal{E}^*, \mathcal{D}^*)$ with a security parameter $\eta \in \mathbb{N}$ as follows:

```
$\mathcal{K}^*(1^\eta)$
    $k \xleftarrow{R} \{0,1\}^\eta$;
    Output $k$.

$\mathcal{E}^*_k(x = x_1 x_2 \cdots x_{L(\eta)-1})$
    $r \xleftarrow{R} \{0,1\}^{l(\eta)}$;
    $y \leftarrow (x \circ 1) \oplus F_k(r)$;
    Output $\langle y, r \rangle$.

$\mathcal{D}^*_k(\langle y = y_1 y_2 \cdots y_{L(\eta)}, r \rangle)$
    $x' \leftarrow y \oplus F_k(r)$;
    Output $x'_1 x'_2 \cdots x'_{L(\eta)-1}$.
```

Note that $\text{Plain}_{\Pi^*[\eta]} = \{0,1\}^{L(\eta)-1}$. Also note that $\mathcal{E}^*$ and $\mathcal{D}^*$ can deduce η from k (i.e., $\eta = |k|$).

Π^* can easily be shown to be IND-CPA secure based on the pseudorandomness of F. For a proof, see [GGM86], or simply think of Π^* as a degenerate version of the randomized CTR mode, and rely on [BDJR97]. Using the results of [AR02], it can further be shown to be Type-0. We have that:

Theorem 2.4.3. Π^* *admits a WKA test.*

Proof. Let A be the following algorithm:

```
$A(1^\eta, \langle y = y_1 y_2 \cdots y_{L(\eta)}, r\rangle, k)$
  $x' \leftarrow y \oplus F_k(r)$;
  If $x'_{L(\eta)}$ is 1
    Output 1;
  Otherwise
    Output 0.
```

We show that A is a WKA test for Π^*. (Note that 1^η is redundant as an input to A here, as A takes k as input and $|k| = \eta$.)

Let $\mathcal{P}_1, \mathcal{P}_2$ be probabilistic, polynomial-time algorithms that take $\eta \in \mathbb{N}$ (in unary) as input and output $x \in \text{Plain}_{\Pi[\eta]}$ for large enough η. Then for such η,

$$\begin{aligned}
&\text{Adv}^{\text{wka}}_{\Pi^*[\eta],\mathcal{P}_1[\eta],\mathcal{P}_2[\eta]}(A) \\
&= \Pr[x \xleftarrow{R} \mathcal{P}_1(1^\eta); k \xleftarrow{R} \mathcal{K}^*(1^\eta); c \xleftarrow{R} \mathcal{E}^*_k(x) : A(1^\eta, c, k) = 1] \\
&\quad - \Pr[x \xleftarrow{R} \mathcal{P}_2(1^\eta); k, k' \xleftarrow{R} \mathcal{K}^*(1^\eta); c \xleftarrow{R} \mathcal{E}^*_k(x) : A(1^\eta, c, k') = 1] \\
&= \Pr[x \xleftarrow{R} \mathcal{P}_1(1^\eta); k \xleftarrow{R} \{0,1\}^\eta; r \xleftarrow{R} \{0,1\}^{l(\eta)}; x' \leftarrow (x \circ 1) \oplus F_k(r) \oplus F_k(r) : x'_{L(\eta)} = 1] \\
&\quad - \Pr[x \xleftarrow{R} \mathcal{P}_2(1^\eta); k, k' \xleftarrow{R} \{0,1\}^\eta; r \xleftarrow{R} \{0,1\}^{l(\eta)}; x' \leftarrow (x \circ 1) \oplus F_k(r) \oplus F_{k'}(r) : x'_{L(\eta)} = 1]
\end{aligned}$$

The first term of the last equality above clearly equals 1. Let $q(\eta)$ denote the second term; we bound it here via a reduction to the pseudorandomness of F. Let B be the following oracle algorithm:

```
$B^g(1^\eta)$
    $x \xleftarrow{R} \mathcal{P}_2(1^\eta);$
    $r \xleftarrow{R} \{0,1\}^{l(\eta)};$
    $y \leftarrow (x \circ 1) \oplus g(r);$
    $k' \xleftarrow{R} \{0,1\}^\eta;$
    $b \leftarrow A(1^\eta, \langle y, r\rangle, k');$
    Output $b$.
```

Now:

$$
\begin{aligned}
&\mathrm{Adv}^{\mathrm{prf}}_{F[\eta]}(B)\\
&\quad = \Pr[k \xleftarrow{R} \{0,1\}^\eta : B^{F_k(\cdot)}(1^\eta) = 1] - \Pr[f \xleftarrow{R} \mathrm{Func}^{l(\eta)\to L(\eta)} : B^{f(\cdot)}(1^\eta) = 1]\\
&\quad = \Pr\left[\begin{array}{ll} x \xleftarrow{R} \mathcal{P}_2(1^\eta); k, k' \xleftarrow{R} \{0,1\}^\eta; r \xleftarrow{R} \{0,1\}^{l(\eta)}; & \\ & : x'_{L(\eta)} = 1 \\ x' \leftarrow (x \circ 1) \oplus F_k(r) \oplus F_{k'}(r) & \end{array}\right]\\
&\qquad - \Pr\left[\begin{array}{ll} x \xleftarrow{R} \mathcal{P}_2(1^\eta); f \xleftarrow{R} \mathrm{Func}^{l(\eta)\to L(\eta)}; k' \xleftarrow{R} \{0,1\}^\eta; & \\ & : x'_{L(\eta)} = 1 \\ r \xleftarrow{R} \{0,1\}^{l(\eta)}; x' \leftarrow (x \circ 1) \oplus f(r) \oplus F_{k'}(r) & \end{array}\right]\\
&\quad = q(\eta) - \frac{1}{2},
\end{aligned}
$$

where the second term above is $\frac{1}{2}$ because $x'_{L(\eta)}$ is the outcome of XORing a truely random bit with some other bit. By the pseudorandomness of F, we have that $\mathrm{Adv}^{\mathrm{prf}}_{F[\eta]}(B) \leq \epsilon(\eta)$ where $\epsilon(\eta)$ is a negligible function. Note that it must also be the case that $\mathrm{Adv}^{\mathrm{prf}}_{F[\eta]}(B) \geq -\epsilon(\eta)$ for any A, otherwise an algorithm with an advantage smaller than $-\epsilon(\eta)$ can be converted into an algorithm with an advantage greater than $\epsilon(\eta)$ by flipping its output. We conclude that:

$$\frac{1}{2} - \epsilon(\eta) \leq q(\eta) \leq \frac{1}{2} + \epsilon(\eta).$$

Putting it all together, we get that:

$$\frac{1}{2} - \epsilon(\eta) \leq \mathrm{Adv}^{\mathrm{wka}}_{\Pi^*[\eta],\mathcal{P}_1[\eta],\mathcal{P}_2[\eta]}(A) \leq \frac{1}{2} + \epsilon(\eta),$$

which is a non-negligible quantity. ■

Theorem 2.4.4. Π^* *does not admit an SKA test.*

Proof. Let A be a probabilistic algorithm that runs in time t, a function of the size of its input. Let $A(a_1, a_2, \ldots; w)$ denote the outcome of running A on inputs $a_1, a_2, \ldots$ and randomness w. Note that the length of w is bounded by t.

Let $\mathcal{U}$ be an algorithm that takes $\eta \in \mathbb{N}$ (in unary) as input and outputs a random, uniformly-selected element of $\{0,1\}^{L(\eta)-1}$. We have:

$$\begin{aligned}
&\mathrm{Adv}^{\mathrm{ska}}_{\Pi^*[\eta],\mathcal{U}[\eta],\mathcal{U}[\eta]}(A) \\
&= \Pr\left[\begin{array}{ll} x \xleftarrow{R} \{0,1\}^{L(\eta)-1}; k \xleftarrow{R} \{0,1\}^{\eta}; r \xleftarrow{R} \{0,1\}^{l(\eta)}; & \\ w \xleftarrow{R} \{0,1\}^{t(\eta)}; y \leftarrow (x \circ 1) \oplus F_k(r) & : A(1^\eta, \langle y, r\rangle, k; w) = 1 \end{array}\right] \\
&\quad - \Pr\left[\begin{array}{ll} x \xleftarrow{R} \{0,1\}^{L(\eta)-1}; k, k' \xleftarrow{R} \{0,1\}^{\eta}; r \xleftarrow{R} \{0,1\}^{l(\eta)}; & \\ w \xleftarrow{R} \{0,1\}^{t(\eta)}; y \leftarrow (x \circ 1) \oplus F_k(r) & : A(1^\eta, \langle y, r\rangle, k'; w) = 1 \end{array}\right],
\end{aligned}$$

where t is a polynomial in η.

Let S_1 and $A_1 \subseteq S_1$ denote the sample space and event, respectively, depicted by the first term above. Let S_2 and $A_2 \subseteq S_2$ be defined similarly with respect to the second term.

Let $(x_0, k_0, r_0, w_0) \in A_1$. Note that for any $k \in \{0,1\}^\eta$, if there exists an $x \in \{0,1\}^{L(\eta)-1}$ such that $(x \circ 1) \oplus F_k(r_0) = (x_0 \circ 1) \oplus F_{k_0}(r_0)$, then it must be the case that $(x, k, k_0, r_0, w_0) \in A_2$ (because in this case, A, in the second experiment, runs on the same input and randomness as in the first experiment). This happens when $x \circ 1 = (x_0 \circ 1) \oplus F_{k_0}(r_0) \oplus F_k(r_0)$, which must happen for at least $\left(\frac{1}{2} - \epsilon(\eta)\right) \cdot 2^\eta$ of the keys $k \in \{0,1\}^\eta$; otherwise, an adversary that queries its oracle on r_0, XORs the answer with $(x_0 \circ 1)$ and with $F_{k_0}(r_0)$, and outputs 1 if the last bit of the result is different than 1, 0 otherwise—breaks the pseudorandomness of F.

For a given $(x_0, k_0, r_0, w_0) \in A_1$, we've just described a way of counting at least $\left(\frac{1}{2} - \epsilon(\eta)\right) \cdot 2^\eta$ tuples in A_2. We would like to argue that for a distinct $(x_1, k_1, r_1, w_1) \in A_1$, we would be counting *different* tuples in A_2 by employing the same method. This is clear if $k_1 \neq k_0$ or $r_1 \neq r_0$ or $w_1 \neq w_0$. As for the case that $k_1 = k_0, r_1 = r_0, w_1 = w_0$, we would be double-counting a tuple iff

$$(x_0 \circ 1) \oplus F_{k_0}(r_0) \oplus F_k(r_0) = (x_1 \circ 1) \oplus F_{k_1}(r_1) \oplus F_k(r_1) = (x_1 \circ 1) \oplus F_{k_0}(r_0) \oplus F_k(r_0),$$

which happens iff $x_1 = x_0$.

We conclude that $|A_2| \geq \left(\frac{1}{2} - \epsilon(\eta)\right) \cdot 2^\eta \cdot |A_1|$. We also know that $|S_2| = 2^\eta \cdot |S_1|$. Therefore:

$$\mathrm{Adv}^{\mathrm{ska}}_{\Pi^*[\eta],\mathcal{U}[\eta],\mathcal{U}[\eta]}(A) = \frac{|A_1|}{|S_1|} - \frac{|A_2|}{|S_2|} \leq \left(\frac{1}{2} + \epsilon(\eta)\right) \cdot \frac{|A_1|}{|S_1|} \leq \frac{1}{2} + \epsilon(\eta),$$

which is *not* negligibly close to 1. ■

Finally, we note that our construction can be easily generalized to one that admits a WKA test with an advantage *as small as desired*, as follows. For any

$c \in \mathbb{N}^+$, let Π_c^* be a variation on Π^* that adds the bit 1 with probability $\frac{1}{2} + \frac{1}{2^c}$, 0 with probability $\frac{1}{2} - \frac{1}{2^c}$, as redundancy upon encryption (instead of the fixed 1). Our proofs easily extend to show that Π_c^* admits a WKA test with advantage at least $\frac{1}{2^c} - \epsilon(\eta)$.

Chapter 3

The Efficiency of Generic Commitments

Here, we demonstrate lower-bounds on the number of times a one-way permutation needs to be invoked (as a "black-box") in order to construct statistically-binding commitments, as outlined in Section 1.2. The chapter is organized as follows. In Section 3.1, we define black-box constructions of statistically-binding schemes based on one-way permutations, and review tools and results used in our proof. In Section 3.2 we prove our lower bounds. In Section 3.3 we compare our bounds with the efficiency of known constructions.

3.1 Definitions and Tools

Our definitions in this chapter use the notion of computational indistinguishability by polynomial-sized circuits; contrast with indistinguishability by probabilistic, polynomial-time algorithms used in Chapter 2, and with the indistinguishability by probabilistic, polynomial-time algorithms taking non-uniform "advice", used in Chapter 4; see discussion in Section 4.1.1.

3.1.1 Preliminaries

Let A^f denote a circuit A with oracle access to the function f. A function $f : \{0,1\}^n \to \{0,1\}^n$ is (S, ε)-*one-way* if for every circuit A of size at most S we have

$$\Pr_x[A^f(f(x)) \in f^{-1}(f(x))] \leq \varepsilon.$$

To reduce the number of parameters, we will call a function S-*hard* if it is $(S, 1/S)$-one way.

Let Π_t denote the set of all permutations over $\{0,1\}^t$. We will rely on the following result:

Theorem 3.1.1 ([GGKT05]). *For sufficiently large t, a random $\pi \in \Pi_t$ is $2^{t/5}$-hard with probability at least $1 - 2^{-2^{t/2}}$.*

Let $a \circ b$ denote the concatenation of strings a and b. For $t < n$, let $\Pi_{t,n}$ denote the subset of Π_n such that $\pi \in \Pi_{t,n}$ iff $\pi(a \circ b) = \hat{\pi}(a) \circ b$ for some $\hat{\pi} \in \Pi_t$ (i.e., the last $n-t$ bits of the input are fixed). A corollary of Theorem 3.1.1 is that if $t = 5 \log S$, then for any $n > t$, a randomly chosen $\pi \in \Pi_{t,n}$ is S-hard with high probability; more formally:

Corollary 3.1.2 ([GGKT05]). *For sufficiently large t and $n > t$, a random $\pi \in \Pi_{t,n}$ is $2^{t/5}$-hard with probability at least $1 - 2^{-2^{t/2}}$.*

We say that two distributions $\mathcal{X}, \mathcal{Y}$ are (S, ε)-*indistinguishable*, and write $\mathcal{X} \overset{(S,\varepsilon)}{\approx} \mathcal{Y}$, if for every circuit Dist of size at most S, we have

$$\left| \Pr_{x \in \mathcal{X}}[\mathsf{Dist}(x) = 1] - \Pr_{x \in \mathcal{Y}}[\mathsf{Dist}(x) = 1] \right| \leq \varepsilon.$$

3.1.2 Commitment Schemes

A *commitment scheme* for m-bit messages is defined by a pair of probabilistic, interactive algorithms $(\mathcal{S}, \mathcal{R})$ representing a *sender* and a *receiver*, respectively. The inputs to $\mathcal{S}$ are a message $M \in \{0,1\}^m$ and a random tape s, while the input to $\mathcal{R}$ is a random tape r. $(\mathcal{S}, \mathcal{R})$ describe a *commitment phase* of the interaction between the parties; call $C = \langle \mathcal{S}(M;s), \mathcal{R}(r) \rangle$ a *commitment to M*. Without loss of generality, we assume that the commitment phase is followed by a generic *decommitment phase*, in which the sender reveals a message a random tape to the receiver; call these a *decommitment.* We say that C as above *can be decommitted to a message M'* if there exists a string s' such that $\langle \mathcal{S}(M';s'), \mathcal{R}(r) \rangle = C$.

Let $\langle \mathcal{S}(M;s), \mathcal{R}^*(r) \rangle$ denote the view of a (possibly malicious) receiver $\mathcal{R}^*$ following an interaction with the sender on the specified inputs; this view consists of the receiver's randomness and the messages it receives from the sender during the interaction (when the receiver makes queries to an oracle, the view also includes the answers it receives from this oracle). For a message M and receiver $\mathcal{R}^*$, define

$$\langle \mathcal{S}(M), \mathcal{R}^* \rangle \stackrel{\text{def}}{=} \left\{ s, r \stackrel{R}{\leftarrow} \{0,1\}^* : \langle \mathcal{S}(M;s), \mathcal{R}^*(r) \rangle \right\};$$

i.e., this denotes the view of $\mathcal{R}^*$ following an interaction with the honest sender who is committing to message M.

We now define the security of a commitment scheme; we deal here with statistically-binding commitments, as reflected by the definitions below.

Definition 3.1.3. Let $(\mathcal{S}, \mathcal{R})$ be a commitment scheme for m-bit messages. We say that $(\mathcal{S}, \mathcal{R})$ is (S_h, ε_h)*-hiding* if for every circuit $\mathcal{R}^*$ of size at most S_h and for all $M_0, M_1 \in \{0,1\}^m$, we have

$$\langle \mathcal{S}(M_0), \mathcal{R}^* \rangle \stackrel{(S_h,\varepsilon_h)}{\approx} \langle \mathcal{S}(M_1), \mathcal{R}^* \rangle.$$

We say that $(\mathcal{S},\mathcal{R})$ is ε_b-*binding* if

$$\Pr_r\left[\begin{array}{c}\exists \text{ distinct } M, M' \in \{0,1\}^m, s, s' \text{ such that} \\ \langle \mathcal{S}(M';s'), \mathcal{R}(r)\rangle = \langle \mathcal{S}(M;s), \mathcal{R}(r)\rangle\end{array}\right] \leq \varepsilon_b.$$

Note that if $(\mathcal{S},\mathcal{R})$ is ε_b-binding then even an all-powerful sender cannot commit to a message M, then later decommit to a different message M', except with probability (at most) ε_b. We say that $(\mathcal{S},\mathcal{R})$ is ε_b-*binding for an honest sender* if for all $M \in \{0,1\}^m$, we have

$$\Pr_{s,r}\left[\begin{array}{c}\exists M' \in \{0,1\}^m \setminus M, s' \text{ such that} \\ \langle \mathcal{S}(M';s'), \mathcal{R}(r)\rangle = \langle \mathcal{S}(M;s), \mathcal{R}(r)\rangle\end{array}\right] \leq \varepsilon_b.$$

Roughly speaking, such a scheme satisfies the following property: if the sender is honest during the commitment phase (and uses a pre-fixed message M and truly random coins s) then the sender cannot later decommit to a different message M' except with probability (at most) ε_b. If $\varepsilon_b = 0$ in either of the above definitions, we say the scheme is *perfectly binding*.

Finally, we say that $(\mathcal{S},\mathcal{R})$ is $(S_h, \varepsilon_h, \varepsilon_b)$-*secure* (resp., *secure for an honest sender*) if $(\mathcal{S},\mathcal{R})$ is (S_h, ε_h)-hiding and ε_b-binding (resp., binding for an honest sender). $\diamondsuit$

We may now define a (weak black-box [RTV04]) construction of a commitment scheme based on one-way permutations.

Definition 3.1.4. A *construction* of a commitment scheme for m-bit messages based on one-way permutations is a pair of oracle algorithms $(\mathcal{S}^{(\cdot)}, \mathcal{R}^{(\cdot)})$ such that, for all $\pi \in \Pi_n$, the resulting $(\mathcal{S}^\pi, \mathcal{R}^\pi)$ is a commitment scheme for m-bit messages.

We say that $(\mathcal{S}^{(\cdot)}, \mathcal{R}^{(\cdot)})$ is a construction which is $(S_p, S_h, \varepsilon_h, \varepsilon_b)$-*secure* (resp., *secure for an honest sender*) if $(\mathcal{S}^{\pi}, \mathcal{R}^{\pi})$ is ε_b-binding (resp., binding for an honest sender) for every $\pi \in \Pi_n$, and furthermore for every $\pi \in \Pi_n$ that is S_p-hard, $(\mathcal{S}^{\pi}, \mathcal{R}^{\pi})$ is (S_h, ε_h)-hiding. ◊

3.1.3 Pairwise-Independent Function Families

Let H be a family of functions mapping m-bit strings to m'-bit strings. We assume that the following can be done in time polynomial in m: (1) selecting a function $h \in H$ uniformly at random; (2) given $h \in H$ and $x \in \{0,1\}^m$, evaluating $h(x)$; and (3) given h^*, deciding whether $h^* \in H$ or not. We say that H is a *pairwise-independent hash family* [CW79] if for any distinct $x_1, x_2 \in \{0,1\}^m$ and any $y_1, y_2 \in \{0,1\}^{m'}$ we have:

$$\Pr_{h \in H}[h(x_1) = y_1 \wedge h(x_2) = y_2] = 2^{-2m'}.$$

Constructions satisfying the above requirements are well known.

3.2 Lower Bounds on the Efficiency of Generic Commitment

Let $(\mathcal{S}^{(\cdot)}, \mathcal{R}^{(\cdot)})$ be an $(S_p, S_h, \varepsilon_h, \varepsilon_b)$-secure construction of a commitment scheme for m-bit messages based on one-way permutations. For $\varepsilon_b > 0$, we prove that unless $\mathcal{S}$ and $\mathcal{R}$ (combined) make $\Omega\left((m - \log(1 + 2^m \cdot \varepsilon_b))/\log S_p\right)$ queries to their oracle, there exists (constructively) a commitment scheme $(\bar{\mathcal{S}}, \bar{\mathcal{R}})$ secure for an honest sender which does *not* require any oracle access at all (i.e., the scheme is secure unconditionally). For $\varepsilon_b = 0$, we show a similar result but where the implication holds unless $\mathcal{S}$ alone makes $\Omega\left(m/\log S_p\right)$ queries to its oracle. In either case, by applying a result of Impagliazzo and Luby [IL89] (cf. also Lemma 3.2.1 below), this implies the unconditional existence of a one-way function, which in turn can be used to give an unconditional construction of a commitment scheme [Nao91].

We begin with an informal discussion of the key ideas behind our proof, focusing for ease of exposition on the case where $(\mathcal{S}^{(\cdot)}, \mathcal{R}^{(\cdot)})$ is perfectly binding. As in [GGKT05], our starting point is that a random $\pi \in \Pi_{t,n}$ (for $t = \Theta(\log S_p)$) is S_p-hard with all but negligible probability (cf. Corollary 3.1.2). Consider the non-interactive scheme $(\mathcal{S}', \mathcal{R}')$ in which $\mathcal{S}'$ locally runs $(\mathcal{S}^{(\cdot)}, \mathcal{R}^{(\cdot)})$, simulating a random $\pi \in \Pi_{t,n}$ for $\mathcal{S}, \mathcal{R}$,[1] and then sends the resulting view of $\mathcal{R}$ to $\mathcal{R}'$.

It is quite straightforward to show that $(\mathcal{S}', \mathcal{R}')$ still satisfies hiding. Binding, however, may not necessarily hold (even when $\mathcal{S}'$ is honest during the commitment phase). To see the issue, assume $\mathcal{S}'$ commits to a message M using coins s for $\mathcal{S}^{(\cdot)}$, coins r for $\mathcal{R}^{(\cdot)}$, and coins y to simulate the permutation. Let C denote the resulting view of $\mathcal{R}$, and let P denote the set of t-bit query/answer prefixes made by $\mathcal{S}$ during the computation. To claim binding, we would need to argue that there does not exist a message $M' \neq M$ along with coins s', y', with an associated set of query/answer prefixes P', that produce an identical view C (note that the coins r are fixed, since r is explicit in C). The most we can claim, though, is that this is true *as long as* $P' = P$, since binding is only guaranteed to hold when the permutation π is fixed, but not when the sender can "change" the permutation after the fact.

[1]This can be done easily by selecting random t-bit answer-prefixes for any new t-bit query-prefixes, as needed; see details in the proof of Theorem 3.2.2.

What we can show is that a weaker form of (honest sender) binding holds for $(\mathcal{S}', \mathcal{R}')$. Observe that for any possible P' (as defined above), there is at most *one* message M' to which the sender can successfully decommit by sending M', s', y' with associated query/answer set P'; this is because $(\mathcal{S}^{(\cdot)}, \mathcal{R}^{(\cdot)})$ is perfectly binding for any fixed permutation. But this implies that there are at most $2^{2t|P'|} = 2^{2tq}$ different messages to which the sender can successfully decommit, where q is the total number of queries made by $\mathcal{S}$ (note that the oracle queries/answers of $\mathcal{R}$ are already fixed by the view C). Although this clearly violates binding, it does limit the space of possible messages to which the sender can decommit, as long as $2^{2tq} < 2^m$.

We now show how to "bootstrap" from the weak form of binding achieved by $(\mathcal{S}', \mathcal{R}')$ to construct a non-interactive scheme $(\bar{\mathcal{S}}, \bar{\mathcal{R}})$ that achieves "full" binding (for an honest sender) with noticeable probability. Sender $\bar{\mathcal{S}}$, on input a message M, proceeds as follows: it first chooses a function h uniformly at random from a pairwise-independent hash family mapping m-bit strings to m-bit strings. It then computes the views $C_1 = \mathcal{S}'(M)$, $C_2 = \mathcal{S}'(h(M))$, and sends $C_1 \circ C_2 \circ h$ to $\bar{\mathcal{R}}$. Hiding for this scheme follows easily via a standard hybrid argument and relying on the fact that $(\mathcal{S}', \mathcal{R}')$ is hiding. As for binding (for an honest sender), we have already seen that C_1 can be decommitted to a set S_1 of at most $2^{2tq} < 2^m$ different messages, and similarly C_2 can be decommitted to a set S_2 of at most 2^{2tq} different messages. For binding not to hold, there must exist an $M' \neq M$ with $M' \in S_1$ and $h(M') \in S_2$. Using the pairwise-independence of h, we can argue that this occurs with only "small" probability over choice of h. Thus, binding for $(\bar{\mathcal{S}}, \bar{\mathcal{R}})$ (for an honest sender) holds with noticeable probability.

3.2.1 Honest-Sender Commitment Implies One-Way Functions

We begin by showing that the existence of a commitment scheme secure for honest senders implies the existence of a one-way function. Although the result can be derived from [IL89], we give a simple and more direct proof here.

Lemma 3.2.1. *Let $(\mathcal{S}, \mathcal{R})$ be a commitment scheme for m-bit messages which is $(S_h, \varepsilon_h, \varepsilon_b)$-secure for an honest sender. Let $S_{\mathcal{S}}, S_{\mathcal{R}}$ be the sizes of the circuits computing $\mathcal{S}, \mathcal{R}$, respectively. Then there exists an $(S_h - S_{\mathcal{S}} + S_{\mathcal{R}} - O(m), \varepsilon_h + 2\varepsilon_b)$-one-way function.*

Proof. Let $S^* = S_h - S_{\mathcal{S}} + S_{\mathcal{R}} - O(m)$ and $\epsilon^* = \varepsilon_h + 2\varepsilon_b$. Define a function f via $f(M, s, r) \stackrel{\text{def}}{=} \langle \mathcal{S}(M; s), \mathcal{R}(r) \rangle$. We claim that f is (S^*, ϵ^*)-one-way. Assume the contrary. Then there exists a circuit B of size at most S^* such that

$$\mathsf{Succ}^{\mathsf{owf}}_{B,f} \stackrel{\text{def}}{=} \Pr_{M,s,r}[B(f(M, s, r)) \in f^{-1}(f(M, s, r))] > \varepsilon^*.$$

We use B to construct a circuit A that violates the hiding property of $(\mathcal{S}, \mathcal{R})$. On input (M_0, M_1, C), where C is either a commitment to M_0 or M_1, A computes $(M', s', r') \leftarrow B(C)$ and checks whether $f(M', s', r') \stackrel{?}{=} C$ and whether $M' \stackrel{?}{=} M_0$. If both hold, A outputs 0; otherwise, it outputs 1. Note that $|A| = |B| + S_{\mathcal{S}} + S_{\mathcal{R}} + O(m) \leq S_h$.

Let $\mathsf{Bad} \stackrel{\text{def}}{=} \{(M, s, r) \mid \exists M' \neq M, s' : \langle \mathcal{S}(M; s), \mathcal{R}(r) \rangle = \langle \mathcal{S}(M'; s'), \mathcal{R}(r) \rangle\}$. In what follows, note that if $(M', s', r') \in f^{-1}(f(M, s, r))$ then $r' = r$, as r is included in the receiver's view. We have:

$$\Pr_{\substack{M_0,M_1\\C\in\langle\mathcal{S}(M_0),\mathcal{R}\rangle}}[A(M_0,M_1,C)=0]$$

$$\begin{aligned}
&= \Pr_{M_0,s,r}\left[\begin{array}{c}(M',s',r')\leftarrow B(f(M_0,s,r)):\\ (M',s',r')\in f^{-1}(f(M_0,s,r))\ \bigwedge\ M'=M_0\end{array}\right]\\
&\geq \Pr_{M_0,s,r}\left[\begin{array}{c}(M',s',r')\leftarrow B(f(M_0,s,r)):\\ (M',s',r')\in f^{-1}(f(M_0,s,r))\ \bigwedge\ (M_0,s,r)\notin \mathsf{Bad}\end{array}\right]\\
&= \Pr_{\substack{M_0,M_1\\s,r}}\left[\begin{array}{c}(M',s',r')\leftarrow B(f(M_0,s,r)):\\ (M',s',r')\in f^{-1}(f(M_0,s,r))\end{array}\right]\\
&\quad -\Pr_{\substack{M_0,M_1\\s,r}}\left[\begin{array}{c}(M',s',r')\leftarrow B(f(M_0,s,r)):\\ (M',s',r')\in f^{-1}(f(M_0,s,r))\bigwedge(M_0,s,r)\in \mathsf{Bad}\end{array}\right]\\
&\geq \Pr_{M_0,s,r}\left[\begin{array}{c}(M',s',r')\leftarrow B(f(M_0,s,r)):\\ (M',s',r')\in f^{-1}(f(M_0,s,r))\end{array}\right]-\Pr_{M_0,s,r}[(M_0,s,r)\in \mathsf{Bad}]\\
&\geq \mathsf{Succ}^{\mathsf{owf}}_{B,f}-\varepsilon_b\\
&= \varepsilon_h+\varepsilon_b.
\end{aligned}$$

Furthermore, we have:

$$\begin{aligned}
&\Pr_{\substack{M_0,M_1\\ C\in\langle\mathcal{S}(M_1),\mathcal{R}\rangle}}[A(M_0,M_1,C)=0] \\
&\quad= \Pr_{\substack{M_0,M_1\\ s,r}}\left[\begin{array}{c}(M',s',r')\leftarrow B(f(M_1,s,r)):\\ (M',s',r')\in f^{-1}(f(M_1,s,r))\bigwedge M'=M_0\end{array}\right] \\
&\quad\le \Pr_{\substack{M_0,M_1\\ s,r}}\left[\begin{array}{c}(M',s',r')\leftarrow B(f(M_1,s,r)):\\ (M',s',r')\in f^{-1}(f(M_1,s,r))\bigwedge(M_1,s,r)\in\mathsf{Bad}\end{array}\right] \\
&\quad\le \Pr_{M_1,s,r}[(M_1,s,r)\in\mathsf{Bad}] \\
&\quad\le \varepsilon_b.
\end{aligned}$$

Putting everything together, we have:

$$\left|\Pr_{\substack{M_0,M_1\\ C\in\langle\mathcal{S}(M_0),\mathcal{R}\rangle}}[A(M_0,M_1,C)=0]-\Pr_{\substack{M_0,M_1\\ C\in\langle\mathcal{S}(M_1),\mathcal{R}\rangle}}[A(M_0,M_1,C)=0]\right|>\varepsilon_h.$$

But this implies that there exist two messages M_0, M_1 for which A can distinguish $\langle\mathcal{S}(M_0),\mathcal{R}\rangle$ from $\langle\mathcal{S}(M_1),\mathcal{R}\rangle$ with probability greater than ε_h, contradicting the hiding of $(\mathcal{S},\mathcal{R})$. ■

3.2.2 Lower Bound

We now formalize the intuition that was discussed earlier. We remark that the proof below is not quite as straightforward as the intuition would suggest, since some technical work is required to deal with the case of statistical (as opposed than perfect) binding.

Theorem 3.2.2. *Let $(\mathcal{S}^{(\cdot)},\mathcal{R}^{(\cdot)})$ be an $(S_p,S_h,\varepsilon_h,\varepsilon_b)$-secure construction of a commitment scheme for m-bit messages that expects an oracle $\pi\in\Pi_n$. Let t =*

$5 \log S_p$. Assume $\varepsilon_h \leq 1/8 - 2^{-S_p}$. If $\varepsilon_b > 0$ and $\mathcal{S}$ and $\mathcal{R}$ make a total of $q \leq (m - 2 - \log(1 + 2^{m+1} \cdot \varepsilon_b))/4t$ queries to their oracle, or if $\varepsilon_b = 0$ and $\mathcal{S}$ makes $q_{\mathcal{S}} \leq (m-2)/4t$ queries to its oracle, then there exists a commitment scheme for m-bit messages which is $(S_h, 1/4, 1/4)$-secure for an honest sender (without access to any oracle).

Applying Lemma 3.2.1, this implies the existence of a one-way function (without access to any oracle).

Proof. We construct non-interactive commitment scheme $(\bar{\mathcal{S}}, \bar{\mathcal{R}})$ for m-bit messages, following the intuition outlined earlier. The construction makes use of a procedure $\mathcal{SIM}$ that simulates a random permutation in $\Pi_{t,n}$ as follows: $\mathcal{SIM}$ maintains a list L which is initially empty. To respond to a query $a \circ a'$, where $|a| = t$ and $|a'| = n-t$, procedure $\mathcal{SIM}$ first checks whether there exists a value b such that $(a, b) \in L$. If so, $\mathcal{SIM}$ returns $b \circ a'$. Otherwise, it picks $b \in \{0,1\}^t \setminus \left\{\hat{b} \mid \exists \hat{a} : (\hat{a}, \hat{b}) \in L\right\}$ uniformly at random, adds (a, b) to L, and returns $b \circ a'$. We let $\mathcal{SIM}_y$ denote an execution of $\mathcal{SIM}$ using random coins y.

Let H be a pairwise-independent family of functions from m-bit strings to m-bit strings. Define $\bar{\mathcal{S}}$ as follows. On input a message $M \in \{0,1\}^m$, $\bar{\mathcal{S}}$ chooses uniformly at random $h \in H$ and values $s_1, r_1, y_1, s_2, r_2, y_2$. It then computes $C_1 = \langle \mathcal{S}^{\mathcal{SIM}_{y_1}}(M; s_1), \mathcal{R}^{\mathcal{SIM}_{y_1}}(r_1) \rangle$ and $C_2 = \langle \mathcal{S}^{\mathcal{SIM}_{y_2}}(h(M); s_2), \mathcal{R}^{\mathcal{SIM}_{y_2}}(r_2) \rangle$, and outputs $C_1 \circ C_2 \circ h$.[2] Decommitment, as usual, is done by having $\bar{\mathcal{S}}$ reveal M and

[2]The permutations simulated by $\mathcal{SIM}$ in the computations of C_1, C_2 will, in general, be different. The theorem can be strengthened (improving the bound on ε_h) by having $\mathcal{SIM}$ provide a consistent simulation for both computations. We forgo this for simplicity.

all the random coins used during the commitment phase. We claim that $(\bar{\mathcal{S}}, \bar{\mathcal{R}})$ is $(S_h, 1/4, 1/4)$-secure for an honest sender. This follows from the following two lemmata.

Lemma 3.2.3. *$(\bar{\mathcal{S}}, \bar{\mathcal{R}})$ is $(S_h, 1/4)$-hiding.*

Proof (of lemma). The hiding property of $(\mathcal{S}^{(\cdot)}, \mathcal{R}^{(\cdot)})$ guarantees that for any $\pi \in \Pi_n$ that is S_p-hard, for any circuit B of size at most S_h, and for any messages $M_0, M_1 \in \{0,1\}^m$, we have

$$\left| \Pr_{C \in \langle \mathcal{S}^{\pi}(M_0), \mathcal{R}^{\pi} \rangle}[B(C) = 0] - \Pr_{C \in \langle \mathcal{S}^{\pi}(M_1), \mathcal{R}^{\pi} \rangle}[B(C) = 0] \right| \leq \varepsilon_h.$$

A straightforward hybrid argument shows that for any $\pi_1, \pi_2 \in \Pi_n$ that are S_p-hard, for any circuit B of size at most S_h, and for any $M_0, M_1 \in \{0,1\}^m$, we have

$$\left| \Pr_{\substack{h \in H \\ C_1 \in \langle \mathcal{S}^{\pi_1}(M_0), \mathcal{R}^{\pi_1} \rangle \\ C_2 \in \langle \mathcal{S}^{\pi_2}(h(M_0)), \mathcal{R}^{\pi_2} \rangle}}[B(C_1 \circ C_2 \circ h) = 0] - \Pr_{\substack{h \in H \\ C_1 \in \langle \mathcal{S}^{\pi_1}(M_1), \mathcal{R}^{\pi_1} \rangle \\ C_2 \in \langle \mathcal{S}^{\pi_2}(h(M_1)), \mathcal{R}^{\pi_2} \rangle}}[B(C_1 \circ C_2 \circ h) = 0] \right| \leq 2\varepsilon_h.$$

Corollary 3.1.2 shows that a random permutation $\pi \in \Pi_{t,n}$ is S_p-hard except with probability at most $2^{-S_p^{5/2}} \leq 2^{-S_p}$. Using a union bound and a simple averaging argument, we see that for any circuit B of size at most S_h and for any $M_0, M_1 \in \{0,1\}^m$,

$$\left| \Pr_{\substack{\pi_1, \pi_2 \in \Pi_{t,n} \\ h \in H \\ C_1 \in \langle \mathcal{S}^{\pi_1}(M_0), \mathcal{R}^{\pi_1} \rangle \\ C_2 \in \langle \mathcal{S}^{\pi_2}(h(M_0)), \mathcal{R}^{\pi_2} \rangle}}[B(C_1 \circ C_2 \circ h) = 0] - \Pr_{\substack{\pi_1, \pi_2 \in \Pi_{t,n} \\ h \in H \\ C_1 \in \langle \mathcal{S}^{\pi_1}(M_1), \mathcal{R}^{\pi_1} \rangle \\ C_2 \in \langle \mathcal{S}^{\pi_2}(h(M_1)), \mathcal{R}^{\pi_2} \rangle}}[B(C_1 \circ C_2 \circ h) = 0] \right| \leq 2\varepsilon_h + 2^{1-S_p}.$$

Since $\mathcal{SIM}$ perfectly simulates a random $\pi \in \Pi_{t,n}$, we have

$$\left| \Pr_{\substack{y_1, y_2 \\ h \in H \\ C_1 \in \langle \mathcal{S}^{\mathcal{SIM}_{y_1}}(M_0), \mathcal{R}^{\mathcal{SIM}_{y_1}} \rangle \\ C_2 \in \langle \mathcal{S}^{\mathcal{SIM}_{y_2}}(h(M_0)), \mathcal{R}^{\mathcal{SIM}_{y_2}} \rangle}}[B(C_1 \circ C_2 \circ h) = 0] - \Pr_{\substack{y_1, y_2 \\ h \in H \\ C_1 \in \langle \mathcal{S}^{\mathcal{SIM}_{y_1}}(M_1), \mathcal{R}^{\mathcal{SIM}_{y_1}} \rangle \\ C_2 \in \langle \mathcal{S}^{\mathcal{SIM}_{y_2}}(h(M_1)), \mathcal{R}^{\mathcal{SIM}_{y_2}} \rangle}}[B(C_1 \circ C_2 \circ h) = 0] \right|$$
$$\leq 2\varepsilon_h + 2^{1-S_p}.$$

But that precisely means that

$$\left| \Pr_{C \in \langle \bar{\mathcal{S}}(M_0), \mathcal{R}^* \rangle} [B(C) = 0] - \Pr_{C \in \langle \bar{\mathcal{S}}(M_1), \mathcal{R}^* \rangle} [B(C) = 0] \right| \leq 2\varepsilon_h + 2^{1-S_p} \leq 1/4$$

for any $\mathcal{R}^*$ and any circuit B of size at most S_h, where the last inequality uses the assumption that $\varepsilon_h \leq 1/8 - 2^{-S_p}$. The hiding property therefore holds as claimed.

$\boxtimes$

Lemma 3.2.4. $(\bar{\mathcal{S}}, \bar{\mathcal{R}})$ *is* $1/4$*-binding for an honest sender.*

Proof (of lemma). For ease of notation, let

$$\mathsf{Com}(M, s, r, y) \stackrel{\text{def}}{=} \langle \mathcal{S}^{\mathcal{SIM}_y}(M; s), \mathcal{R}^{\mathcal{SIM}_y}(r) \rangle.$$

Fix an arbitrary $M \in \{0,1\}^m$. We are interested in the following probability:

$$\begin{aligned}
\mathsf{NoBind} &\stackrel{\text{def}}{=} \Pr_{\bar{s}} \left[\begin{array}{c} \exists M' \in \{0,1\}^m \setminus M, \bar{s}' \text{ such that} \\ \langle \bar{\mathcal{S}}(M'; \bar{s}'), \bar{\mathcal{R}} \rangle = \langle \bar{\mathcal{S}}(M; \bar{s}), \bar{\mathcal{R}} \rangle \end{array} \right] \\
&= \Pr_{\substack{h \in H \\ s_1, r_1, y_1 \\ s_2, r_2, y_2}} \left[\begin{array}{c} \exists M' \in \{0,1\}^m \setminus M, h', s_1', r_1', y_1', s_2', r_2', y_2' \text{ such that} \\ \mathsf{Com}(M', s_1', r_1', y_1') \circ \mathsf{Com}(h'(M'), s_2', r_2', y_2') \circ h' \\ = \mathsf{Com}(M, s_1, r_1, y_1) \circ \mathsf{Com}(h(M), s_2, r_2, y_2) \circ h \end{array} \right] \\
&= \Pr_{\substack{h \in H \\ s_1, r_1, y_1 \\ s_2, r_2, y_2}} \left[\begin{array}{c} \exists M' \in \{0,1\}^m \setminus M, s_1', y_1', s_2', y_2' \text{ such that} \\ \mathsf{Com}(M', s_1', r_1, y_1') = \mathsf{Com}(M, s_1, r_1, y_1) \bigwedge \\ \mathsf{Com}(h(M'), s_2', r_2, y_2') = \mathsf{Com}(h(M), s_2, r_2, y_2) \end{array} \right],
\end{aligned}$$

where in the last equality we use the fact that h', r_1', r_2' and h, r_1, r_2 are explicit in the view of $\bar{\mathcal{R}}$. Letting

$$\mathsf{Decom}(M, s, r, y) \stackrel{\text{def}}{=} \left\{ M' \in \{0,1\}^m \;\middle|\; \begin{array}{c} \exists s', y' \text{ such that} \\ \mathsf{Com}(M', s', r, y') = \mathsf{Com}(M, s, r, y) \end{array} \right\},$$

we may write:

$$\mathsf{NoBind} = \Pr_{\substack{h\in H\\ s_1,r_1,y_1\\ s_2,r_2,y_2}}\left[\begin{array}{c}\exists M' \in \{0,1\}^m \setminus M \text{ such that}\\ M' \in \mathsf{Decom}(M, s_1, r_1, y_1) \bigwedge\\ h(M') \in \mathsf{Decom}(h(M), s_2, r_2, y_2)\end{array}\right].$$

Let $q_{\mathcal{S}}$ (resp., $q_{\mathcal{R}}$) denote the number of queries made by $\mathcal{S}$ (resp., $\mathcal{R}$) to its oracle[3], and let $q = q_{\mathcal{S}} + q_{\mathcal{R}}$. For any integer q^*, let $\mathsf{Perm}_t^{q^*}$ denote the set of "partial permutations" of size q^* over t-bit strings; formally, $\mathsf{Perm}_t^{q^*}$ contains all sets $P \subseteq \{0,1\}^t \times \{0,1\}^t$ such that P contains exactly q^* tuples and such that for all a there exists at most one b with $(a,b) \in P$ and at most one b' such that$(b', a) \in P$. Let $\mathsf{queries}(M, s, r, y) \in \mathsf{Perm}_t^q$ denote the set of query/answer prefixes made by either $\mathcal{S}$ or $\mathcal{R}$ to $\mathcal{SIM}$ during the computation of $\mathsf{Com}(M, s, r, y)$ (i.e., $(a, b) \in \mathsf{queries}(M, s, r, y)$ iff an oracle query $a \circ a'$, by either $\mathcal{S}$ or $\mathcal{R}$, is answered by $\mathcal{SIM}$ with $b \circ a'$ during the computation of $\mathsf{Com}(M, s, r, y)$). Define $\mathsf{queries}_{\mathcal{S}}(M, s, r, y)$ (resp., $\mathsf{queries}_{\mathcal{R}}(M, s, r, y)$) similarly, where this refers exclusively to queries made by $\mathcal{S}$ (resp., $\mathcal{R}$).

Define r as *good for* $P \in \mathsf{Perm}_t^q$ if there do not exist distinct M', M'', along with s', s'', y', y'', such that

- $\mathsf{Com}(M', s', r, y') = \mathsf{Com}(M'', s'', r, y'')$; and
- $\mathsf{queries}(M', s', r, y') = \mathsf{queries}(M'', s'', r, y'') = P$.

Say r is *good* if it is good for all $P \in \mathsf{Perm}_t^q$.

[3]Without loss of generality, we will assume that exactly $q_{\mathcal{S}}$ (resp., $q_{\mathcal{R}}$) queries are always made.

We first observe that for a good r, the set $\mathsf{Decom}(M, s, r, y)$ contains at most $|\mathsf{Perm}_t^{qs}| < 2^{2tqs}$ messages. Otherwise, by the pigeonhole principle, there exists a $P_{\mathcal{S}} \in \mathsf{Perm}_t^{qs}$ and distinct messages $M', M'' \in \mathsf{Decom}(M, s, r, y)$, along with s', s'', y', y'', such that $\mathsf{Com}(M', s', r, y') = \mathsf{Com}(M, s, r, y) = \mathsf{Com}(M'', s'', r, y'')$ and $\mathsf{queries}_{\mathcal{S}}(M', s', r, y') = \mathsf{queries}_{\mathcal{S}}(M'', s'', r, y'') = P_{\mathcal{S}}$. Notice also that $\mathsf{queries}_{\mathcal{R}}(M', s', r, y') = \mathsf{queries}_{\mathcal{R}}(M, s, r, y) = \mathsf{queries}_{\mathcal{R}}(M'', s'', r, y'')$, as these queries are explicit in the receiver's views $\mathsf{Com}(M', s', r, y') = \mathsf{Com}(M, s, r, y) = \mathsf{Com}(M'', s'', r, y'')$. But then r is not good for $P \stackrel{\text{def}}{=} P_{\mathcal{S}} \cup \mathsf{queries}_{\mathcal{R}}(M, s, r, y)$, contradicting the assumption that r is good for all $P \in \mathsf{Perm}_t^q$.

Fix some $P \in \mathsf{Perm}_t^q$, and let π_P denote an arbitrary extension of P to a permutation in $\Pi_{t,n}$ (in the natural way). We have

$$\begin{aligned}
\Pr_r[r \text{ is not good for } P] &= \Pr_r\left[\begin{array}{c} \exists \text{ distinct } M', M'', \exists s', s'', y', y'' \text{ such that} \\ \mathsf{Com}(M', s', r, y') = \mathsf{Com}(M'', s'', r, y'') \bigwedge \\ \mathsf{queries}(M', s', r, y') = \mathsf{queries}(M'', s'', r, y'') = P \end{array}\right] \\
&\leq \Pr_r\left[\begin{array}{c} \exists \text{ distinct } M', M'', \exists s', s'' \text{ such that} \\ \langle \mathcal{S}^{\pi_P}(M'; s'), \mathcal{R}^{\pi_P}(r)\rangle = \langle \mathcal{S}^{\pi_P}(M''; s''), \mathcal{R}^{\pi_P}(r)\rangle \end{array}\right] \\
&\leq \varepsilon_b\,,
\end{aligned}$$

by the binding property of $(\mathcal{S}^{(\cdot)}, \mathcal{R}^{(\cdot)})$. Applying a union bound over all elements of Perm_t^q, we obtain:

$$\Pr_r[r \text{ is not good}] < 2^{2tq} \cdot \varepsilon_b.$$

We proceed to bound NoBind. We have:

$$\mathsf{NoBind} \leq \underbrace{\Pr_{\substack{h \in H \\ s_1, r_1, y_1 \\ s_2, r_2, y_2}} \left[\begin{array}{c|c} \exists M' \in \{0,1\}^m \setminus M \ \text{ such that} & \\ M' \in \mathsf{Decom}(M, s_1, r_1, y_1) & r_1, r_2 \text{ good} \\ h(M') \in \mathsf{Decom}(h(M), s_2, r_2, y_2) & \end{array} \right]}_{\mathsf{LeftTerm}} + 2^{2tq+1} \cdot \varepsilon_b,$$

where the right term above represents an upper-bound on the probability that either r_1 or r_2 is not good. Continuing with the left term, we have

$$\begin{aligned} \mathsf{LeftTerm} &= \sum_{M_2 \in \{0,1\}^m} \Pr_{\substack{h \in H \\ s_1, r_1, y_1 \\ s_2, r_2, y_2}} \left[\begin{array}{c|c} \exists M' \in \mathsf{Decom}(M, s_1, r_1, y_1) \setminus M, & \\ \exists M_2' \in \mathsf{Decom}(M_2, s_1, r_1, y_1) \ \text{ such that} & r_1, r_2 \text{ good} \\ h(M) = M_2 \bigwedge h(M') = M_2' & \end{array} \right] \\ &= \sum_{M_2 \in \{0,1\}^m} \left(2^{-2m} \cdot \max_{\substack{s_1, \text{ good } r_1, y_1 \\ s_2, \text{ good } r_2, y_2}} \{ |\mathsf{Decom}(M, s_1, r_1, y_1)| \cdot |\mathsf{Decom}(M_2, s_2, r_2, y_2)| \} \right), \end{aligned}$$

using the pairwise independence of H. Applying the bound on the size of $\mathsf{Decom}(M, s, r, y)$ when r is good, we obtain

$$\mathsf{LeftTerm} \leq 2^{-2m} \cdot 2^m \cdot 2^{4tq_{\mathcal{S}}} = 2^{4tq_{\mathcal{S}} - m}.$$

Putting everything together, we have

$$\mathsf{NoBind} \leq 2^{4tq_{\mathcal{S}} - m} + 2^{2tq+1} \cdot \varepsilon_b \,.$$

If $\varepsilon_b = 0$ and $q_{\mathcal{S}} \leq (m-2)/4t$, it is easy to see that $\mathsf{NoBind} \leq 1/4$. When $\varepsilon_b > 0$ and $q \leq (m - 2 - \log(1 + 2^{m+1} \cdot \varepsilon_b))/4t$, we may observe that $2^{4tq_{\mathcal{S}} - m} + 2^{2tq+1} \cdot \varepsilon_b \leq 2^{4tq} \cdot (2^{-m} + 2\varepsilon_b)$ and hence $\mathsf{NoBind} \leq 1/4$ in this case as well. The claim follows. ⊠

This completes the proof of the theorem. ■

3.3 Upper Bounds on the Efficiency of Generic Commitment

Here, we briefly describe upper bounds on the efficiency of black-box constructions of commitment schemes based on one-way permutations and compare them with our lower bounds.

3.3.1 Perfectly-Binding Commitment

A perfectly-binding commitment scheme can be constructed from one-way permutations using the approach of Blum [Blu82] along with the Goldreich-Levin hard-core function paradigm [GL89]. Specifically, let $h : \{0,1\}^n \to \{0,1\}^\ell$ be a *hard-core function* (see [Gol01]) for a one-way permutation $\pi : \{0,1\}^n \to \{0,1\}^n$. To commit to a message $M \in \{0,1\}^m$, the sender first divides M into $t = \lceil m/\ell \rceil$ blocks $N_1, \ldots, N_t$, each of length ℓ. Then, for each block N_i the sender chooses a random $s_i \in \{0,1\}^n$ and sends $\pi(s_i), h(s_i) \oplus N_i$ to the receiver. Since there exists a hard-core function with $\ell = O(\log S)$ for any S-hard π (and large enough n) [GL89] (see also [Gol01, Section 2.5.3]), this construction requires $O(m/\log S)$ invocations of π, matching our bound.

3.3.2 Statistically-Binding Commitment for Single-Bit Messages

Naor [Nao91] showed a construction of a statistically-binding commitment scheme for single-bit messages based on one-way *functions*. Let $G : \{0,1\}^n \to \{0,1\}^{n+k}$ be a pseudorandom generator. The receiver first chooses a random $r \in \{0,1\}^{n+k}$ and sends this value to the other party. The sender then commits to a bit

b as follows: it chooses a random $s \in \{0,1\}^n$ and sends $G(s)$ if $b = 0$ and $G(s) \oplus r$ if $b = 1$. This scheme is binding with $\varepsilon_b < 2^{2n}/2^{n+k} = 2^{n-k}$.

Although a pseudorandom generator can be constructed from one-way functions, we will examine the efficiency of the above scheme when G is based on an S-hard one-way permutation $\pi : \{0,1\}^n \to \{0,1\}^n$ so as to compare the efficiency of the scheme to our bound. In this case, evaluating G requires $O(k/\log S)$ invocations of π [Yao82, BM84, GL89]. Viewing n as fixed, this is $O(\log \varepsilon_b^{-1}/\log S)$ invocations of π (for k polynomial in n).

3.3.3 Statistically-Binding Constructions for Longer Messages

There are a number of ways to extend the Naor scheme described above for the case of m-bit messages. One obvious approach is to simply run the basic Naor scheme in parallel for each bit of the message, having the sender/receiver use the same value r for all these commitments. This gives a scheme which is binding with $\varepsilon_b < 2^{n-k}$ as before, but where the number of invocations of π required is now $O(mk/\log S)$.

A better approach, suggested in [Nao91], is to have the sender use the above idea to commit to an n-bit *seed* s, and then additionally send $G'(s) \oplus M$ (where M is the sender's message and $G' : \{0,1\}^n \to \{0,1\}^m$ is another pseudorandom generator). This is still binding with $\varepsilon_b < 2^{n-k}$ as before; the number of invocations of π required, however, is $O(nk/\log S + (m-n)/\log S)$ which is more efficient than the previous approach when $m > n$.

A third approach, suggested in [Nao91] as well, utilizes asymptotically good error-correcting codes to extend the basic scheme. We present a simpler construction here which achieves the same efficiency and which (to the best of our knowledge) has not appeared before. Let $G : \{0,1\}^n \to \{0,1\}^\ell$ be a pseudorandom generator, where ℓ will be fixed later. The receiver begins by choosing random $r_1, \ldots, r_m \in \{0,1\}^\ell$ and transmitting these to the sender. The sender chooses a random $s \in \{0,1\}^n$ and responds with $\left(\bigoplus_{i:M_i=1} r_i\right) \oplus G(s)$ (where M_i is the i^{th} bit of M). As in the basic Naor scheme, hiding follows easily from the pseudorandomness of G. As for binding, we have

$$\Pr_{r_1,\ldots,r_m}\left[\begin{array}{c} \exists M \neq M', s, s' \text{ such that} \\ \left(\bigoplus_{i:M_i=1} r_i\right) \oplus G(s) = \left(\bigoplus_{i:M'_i=1} r_i\right) \oplus G(s') \end{array}\right]$$

$$\begin{aligned}
&= \Pr_{r_1,\ldots,r_m}\left[\begin{array}{c} \exists M \neq M', s, s' \text{ such that} \\ \bigoplus_{i:M_i \oplus M'_i=1} r_i = G(s) \oplus G(s') \end{array}\right] \\
&= \Pr_{r_1,\ldots,r_m}\left[\begin{array}{c} \exists N \neq 0^m, s, s' \text{ such that} \\ \bigoplus_{i:N_i=1} r_i = G^\pi(s) \oplus G^\pi(s') \end{array}\right] \\
&\leq \sum_{\substack{\hat{s},\hat{s}' \\ \hat{N} \neq 0^m}} \Pr_{r_1,\ldots,r_m}\left[\bigoplus_{i:\hat{N}_i=1} r_i = G^\pi(\hat{s}) \oplus G^\pi(\hat{s}')\right] \\
&< 2^m \cdot 2^{2n} \cdot 2^{-\ell}.
\end{aligned}$$

Setting $\ell = n + m + k$, we obtain a scheme that binds except with probability $\varepsilon_b < 2^{n-k}$ (as previously) and which requires only $O((m+k)/\log S)$ invocations of an S-hard permutation π.

Chapter 4

Universally-Composable Two-Party Computation in Two Rounds

Here, we give a tight characterization of the round complexity of secure two-party computation in the UC framework, as outlined in Section 1.3. The chapter is organized as follows. In Section 4.1, we give an overview the UC framework, and recall tools and assumptions used in our constructions. In Section 4.2, we present a two-round, UC two-party computation protocol for the setting in which parties may speak simultaneously in any given round; and a three-round protocol for the task when parties take turns in transmitting their protocol messages. At the end of the section, we observe that the protocols are round-optimal in their respective communication models. In Section 4.3, we discuss how our results can be applied to obtain a round-optimal UC blind signature scheme. In Section 4.4, we briefly compare the application of our main protocol to evaluating policies on sets of credentials with other approaches to the task.

4.1 Framework, Tools, and Assumptions

4.1.1 Preliminaries

In this chapter, we consider computational-indistinguishability by probabilistic, polynomial-time algorithms taking non-uniform "advice". The notion can easily be shown equivalent to the notion of computational-indistinguishability by polynomial-sized circuit families, used in Chapter 3. It may also be shown to be strictly stronger than the notion of computational-indistinguishability by polynomial-time algorithms used in Chapter 2; see [Gol01, Section 3.2.4] for a discussion.

Let $X = \{X(k,z)\}_{k\in\mathbb{N},z\in\{0,1\}^*}$ denote an ensemble of binary distributions, where $X(k,z)$ represents the output of a probabilistic, polynomial time (PPT) algorithm on a *security parameter* k and advice z (the ensemble may be parameterized by additional variables, and the algorithm may take additional inputs). We say that ensembles X, Y are *computationally indistinguishable*, and write $X \overset{\text{c}}{\approx} Y$, if for any $a \in \mathbb{N}$ there exists $k_a \in \mathbb{N}$ such that for all $k > k_a$, for all z (and for all values any additional variables parameterizing the ensemble may take), we have

$$|\Pr[X(k,z) = 1] - \Pr[Y(k,z) = 1]| < k^{-a}.$$

4.1.2 Universally Composable Security

We consider secure computation within the Universal Composability framework of Canetti [Can01], which we review here, most closely following the treatment of [CLOS02]. Our focus is on the two-party, static corruption setting. We highlight a few features of our definition that are standard but not universal: (1) The real model offers authenticated communication and universal access to a common reference string. Formally, this corresponds to the $(\mathcal{F}_{\text{AUTH}}, \mathcal{F}_{\text{CRS}})$-hybrid model of [Can01]. (2) Message delivery in both the real and ideal models is carried out by the adversary (contrast with [Can01], where messages between the dummy parties and the ideal functionality in the ideal model are delivered immediately). (3) The ideal functionality is not informed of party corruption by the ideal adversary. We make this choice purely to simplify the exposition; our results extend to the more general setting by the same means employed in [CLOS02] (see section 3.3 there).

4.1.2.1 Program and protocol syntax

Following [GMR89, Gol01], we use probabilistic, interactive Turing machines (ITMs) to model programs to be run by parties in our execution environment. Input and output tapes of ITMs are used to model inputs and outputs that are received from and given to other programs running within a party, while communication tapes are used to model messages sent to and received from the network. Hereafter, when referring to a party, we mean the instance of the ITM it is running (when that is clear from the context).

Protocols are specified by a set of ITMs, representing the programs to be run by the participating parties. A protocol may also specify a distribution from which it expects a common reference string (CRS) to be drawn. To simplify the exposition, we assume that all protocols are such that the ITMs read their input tapes only at the onset of their computation (this can easily be achieved by having an ITM copy its input tape onto an internal work tape).

4.1.2.2 Defining the security of protocols

A protocol is said to be secure if its execution in a given real-life setting in the presence of an adversary essentially "emulates" an ideal process capturing the desired task. Below, we define the real-life setting, the ideal process and the notion of protocol emulation.

Protocol execution in the real-life model. Execution in the real-life model involves a protocol π, to be run by parties P_1, P_2; an *adversary* $\mathcal{A}$; and an *environment* $\mathcal{Z}$ with input z. All parties have a security parameter $k \in \mathbb{N}$ and are polynomial in k.

On the onset of an execution, a CRS from the distribution specified by π (if any) is chosen; the parties and the adversary have read access to the CRS throughout the execution. Note that this may be realized in the $\mathcal{F}_{\text{CRS}}$-hybrid model of [Can01]. The execution consists of a sequence of *activations*, where in each activation, a single party (be it $P_1, P_2, \mathcal{A}$ or Z) is running. The environment is activated first. On its first activation, it may write a single message on the input tape of the adversary, who's activated next. The intent is to allow the (static) adversary to corrupt parties on the onset of the execution (on corruption, see below). In each subsequent activation, the environment may read the contents of the output tapes of all uncorrupted parties and the adversary, and may write a message on the input tape of a single uncorrupted party or the adversary (the environment may pass an input to a corrupted party by sending it directly to the adversary, who controls it; see below). Once the activation is complete, the entity whose input tape was written to is activated next.

On its first activation, the adversary may *corrupt* one or both parties (or neither). Upon corruption, the adversary gains access to all the tapes of a corrupted party and may arbitrarily act on its behalf in the future (the corrupted party is never activated). The environment is notified of the identity of the corrupted parties, if any (say, via a message that is added to the output tape of the adversary). Note that corruption here corresponds to the notion of a *static* adversary of [Can01]. Once the first activation is complete, the environment is reactivated. On any subsequent activation, the adversary may read its input tape and the outgoing communication tapes of all uncorrupted parties. It may then *deliver* a message to a party by writing it on the party's incoming communication tape. We make the following restrictions on message delivery: if P_i is not corrupted, then the adversary may deliver a message m from P_i to P_j if and only if m was previously written onto the outgoing communication tape of P_i with P_j as the designated recipient; furthermore, m may only be delivered once. Messages need not be delivered in the order in which they were sent. Note that this models an asynchronous network with authenticated links; formally, this corresponds to the $\mathcal{F}_{\text{AUTH}}$-hybrid model of [Can01]. If the adversary delivered a message to some uncorrupted party in its activation then this party is activated once the activation of the adversary is complete. Otherwise, the environment is activated next.

Once an (uncorrupted) party is activated (either due to an input given by the environment or due to a message delivered by the adversary), it follows its program and may write local outputs on its output tape and *send* messages to other parties by writing them on its outgoing communication tape. Once the activation of the party is complete, the environment is activated.

The protocol execution ends when the environment completes an activation without writing on the input tape of any entity. The output of the execution is the output of the environment. We assume that this output consists of a single bit.

In summary, the order of activations is as follows. The environment is activated first. On its first activation, it activates the adversary, who then returns control to the environment. In subsequent activations, the environment may activate the adversary or an uncorrupted party by writing on an input tape. If the adversary is activated, it may return control to the environment, or it may activate a party by delivering a message to it. After a party is activated, control is always returned to the environment. We stress that at any point, only a single party is activated. Furthermore the environment and the adversary can only activate one other entity (thus only a single input is written by the environment per activation, and likewise the adversary can only deliver a single message per activation).

Let $\mathrm{REAL}_{\pi,\mathcal{A},\mathcal{Z}}(k, z, \bar{r})$ denote the output of the environment $\mathcal{Z}$ when interacting with adversary $\mathcal{A}$ and parties running protocol π on security parameter k, input z and random tapes $\bar{r} = r_{\mathcal{Z}}, r_{\mathcal{A}}, r_1, r_2$ as described above (z and $r_{\mathcal{Z}}$ for $\mathcal{Z}$, $r_{\mathcal{A}}$ for $\mathcal{A}$, r_i for P_i). Let $\mathrm{REAL}_{\pi,\mathcal{A},\mathcal{Z}}(k, z)$ denote the random variable describing $\mathrm{REAL}_{\pi,\mathcal{A},\mathcal{Z}}(k, z, \bar{r})$ when $\bar{r}$ is uniformly chosen. Let $\mathrm{REAL}_{\pi,\mathcal{A},\mathcal{Z}}$ denote the ensemble $\{\mathrm{REAL}_{\pi,\mathcal{A},\mathcal{Z}}(k, z)\}_{k\in\mathbb{N},z\in\{0,1\}^*}$.

The ideal process. The ideal process involves an *ideal functionality* $\mathcal{F}$, an *ideal process adversary (simulator)* $\mathcal{S}$, an environment $\mathcal{Z}$ with input z, and *dummy parties* $\tilde{P}_1, \tilde{P}_2$. $\mathcal{F}$ is modeled as an ITM; $\mathcal{Z}$ and $\mathcal{S}$ have a security parameter $k \in \mathbb{N}$ and are polynomial in k.

As in protocol execution in the real-life model, the ideal process consists of a sequence of activations, where in each activation, a single party (be it $\mathcal{F}, \mathcal{S}, \tilde{P}_1, \tilde{P}_2$ or Z) is running. The environment is activated first. As there, on its first activation, the environment may write a single message on the input tape of the simulator, who's activated next. The intent is to allow the (static) simulator to corrupt parties on the onset of the process. In each subsequent activation, the environment may read the contents of the output tapes of all uncorrupted dummy parties and the simulator, and may write a message on the input tape of a single, uncorrupted dummy party or the simulator (the environment may pass an input to a corrupted dummy party by sending it directly to the simulator, who controls it; see below). Once the activation is complete, the entity whose input tape was written to is activated next.

The dummy parties are fixed and simple ITMs. Whenever a dummy party is activated with an input, it writes it on its outgoing communication tape for the ideal functionality; whenever a dummy party is activated due to the delivery of some message (from the ideal functionality), it copies the message onto its output. At the conclusion of a dummy party's activation, the environment is activated. The communication of the dummy parties is with the ideal functionality only. A message between the two entities comprises of a *header* and *contents*. In this work, as a convention, the first two fields of a message will constitute the header, and the rest the contents. The first field of the header will describe the action being taken, and the second will identify the *session* being run (see also Section 4.1.2.3).

When the ideal functionality is activated, it may read the contents of its incoming communication tape, and may send messages to the dummy parties and the simulator by writing these messages on its outgoing communication tape. Once the activation of the ideal functionality is complete, the environment is activated next.

On its first activation, the simulator may *corrupt* one or both dummy parties (or neither). Upon corruption, the simulator gains access to all the tapes of a corrupted party and may arbitrarily act on its behalf in the future (the corrupted party is never activated). The environment is notified of the identity of the corrupted parties, if any (again, say via a message that is added to the output tape of the simulator). Note that corruption corresponds to the notion of a *static* adversary. Once the first activation is complete, the environment is reactivated. On any subsequent activation, the simulator may read its input tape; the *headers* (but not the contents) of messages on the outgoing communication tape of the ideal functionality intended for uncorrupted dummy parties; the *headers and contents* of messages on the outgoing communication tape of the ideal functionality intended for the simulator or corrupted parties (note that the ideal process allows the simulator to learn output values sent by the ideal functionality to corrupted parties as soon as they are generated); and the *headers* (but not the contents) of messages on the outgoing communication tapes of uncorrupted dummy parties intended for the ideal functionality. The simulator may then *deliver* a message, either from a dummy party to the ideal functionality, or from the ideal functionality to an uncorrupted dummy party, by writing it onto the incoming communication tape of the recipient. As in the execution in the real-life model, messages between uncorrupted dummy parties and the ideal functionality may only be delivered if they appear on the outgoing communication tape of the sender, are intended for the recipient, and have not been delivered before. There is no restriction on the order in which messages are delivered. If the simulator delivered a message to some uncorrupted dummy party or to the ideal functionality in an activation, then that entity is activated once the activation of the simulator is complete. Otherwise, the environment is activated next.

As in the real-life model, the ideal process ends when the environment completes an activation without writing on the input tape of any entity. The output of the protocol is the (single bit) output of the environment.

In summary, the order of activations is as follows. The environment is activated first. On its first activation, it activates the simulator, who then returns control to the environment. In subsequent activations, the environment may activate the simulator or an uncorrupted dummy party by writing on an input tape. If the simulator is activated, it may return control to the environment, or it may activate either a dummy party or the ideal functionality by delivering a message to that entity. After the activation of either an uncorrupted, dummy party or the ideal functionality, control is always returned to the environment.

Let $\mathrm{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}(k, z, \bar{r})$ denote the output of the environment $\mathcal{Z}$ after interacting in the ideal process with simulator $\mathcal{S}$ and the ideal functionality $\mathcal{F}$, on security parameter k, input z, and random input $\bar{r} = r_{\mathcal{Z}}, r_{\mathcal{S}}, r_{\mathcal{F}}$ as described above (z and $r_{\mathcal{Z}}$ for $\mathcal{Z}$; $r_{\mathcal{S}}$ for $\mathcal{S}$; $r_{\mathcal{F}}$ for $\mathcal{F}$). Let $\mathrm{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}(k, z)$ denote the random variable describing $\mathrm{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}(k, z, \bar{r})$ when $\bar{r}$ is chosen uniformly. Let $\mathrm{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$ denote the ensemble $\{\mathrm{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}\}_{k\in\mathbb{N},z\in\{0,1\}^*}$.

We remark that the above definition of the ideal process slightly differs from that of [Can01] in that there, messages between the dummy parties and the ideal functionality are delivered *immediately.* In contrast, in this presentation, following [CLOS02], the delivery is carried out by the simulator. Thus, in both the real and ideal models, *all* message delivery is the responsibility of the adversary alone. We note that our results can easily be stated in the model of "immediate delivery" as defined in [Can01].

UC realizing an ideal functionality (emulation of the ideal process). We say that a protocol π *UC realizes* an ideal functionality $\mathcal{F}$ if for any real-life adversary $\mathcal{A}$ there exists an ideal-process adversary $\mathcal{S}$ such that no environment $\mathcal{Z}$ can tell with non-negligible probability whether it is interacting with $\mathcal{A}$ and parties running π in the real-life process, or with $\mathcal{S}$ and $\mathcal{F}$ in the ideal process. Formally:

Definition 4.1.1. Let $\mathcal{F}$ be an ideal functionality and let π be a two-party protocol. We say that π *UC realizes* $\mathcal{F}$ if for any adversary $\mathcal{A}$ there exists an ideal-process adversary $\mathcal{S}$ such that for any environment $\mathcal{Z}$,

$$\mathrm{REAL}_{\pi,\mathcal{A},\mathcal{Z}} \overset{c}{\approx} \mathrm{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}.$$

$\diamondsuit$

4.1.2.3 The composition theorem

The hybrid model. In order to state the composition theorem, and in particular in order to formalize the notion of a real-life protocol with access to multiple copies of an ideal functionality, the *hybrid model of computation* with access to an ideal-functionality $\mathcal{F}$ (the *$\mathcal{F}$-hybrid model*, in short) is formalized. The model is identical to the real-life model, with the following additions. On top of sending messages to each other, the parties may send messages to and receive messages from an unbounded number of copies of $\mathcal{F}$. Each copy of $\mathcal{F}$ is identified via a unique *session identifier (SID)*; all messages addressed to this copy and all messages sent by this copy carry the corresponding SID.

The communication between the parties and each one of the copies of $\mathcal{F}$ mimics the ideal process. Specifically, when the adversary delivers a message from a party to a copy of $\mathcal{F}$ with a particular SID, that copy of $\mathcal{F}$ is the next entity to be activated. Furthermore, although the adversary in the hybrid model is responsible for delivering the messages between the copies of $\mathcal{F}$ and the parties, it does not have access to the contents of these messages.

The hybrid model does not specify how the SIDs are generated, nor does it specify how parties "agree" on the SID of a certain protocol copy that is to be run by them. These tasks are left to the protocol in the hybrid model. This convention simplifies formulating ideal functionalities and designing protocols that UC realize them, by freeing the functionality from the need to choose the SIDs and guarantee their uniqueness (see [CLOS02] for further discussion).

Let $\mathrm{EXEC}^{\mathcal{F}}_{\pi,\mathcal{A},\mathcal{Z}}(k,z)$ denote the random variable describing the output of the environment $\mathcal{Z}$ on input z, after interacting in the $\mathcal{F}$-hybrid model with protocol π and adversary $\mathcal{A}$, analogously to the definition of $\mathrm{REAL}_{\pi,\mathcal{A},\mathcal{Z}}(k,z)$. (We stress that here π is a hybrid of a real-life protocol with ideal evaluation calls to $\mathcal{F}$.) Let $\mathrm{EXEC}^{\mathcal{F}}_{\pi,\mathcal{A},\mathcal{Z}}$ denote the distribution ensemble $\left\{\mathrm{EXEC}^{\mathcal{F}}_{\pi,\mathcal{A},\mathcal{Z}}\right\}_{k\in\mathbb{N},z\in\{0,1\}^*}$.

Replacing a call to $\mathcal{F}$ with a protocol invocation. Let π be a protocol in the $\mathcal{F}$-hybrid model, and let ρ be a protocol that UC-realizes $\mathcal{F}$ (with respect to some class of adversaries). The *composed protocol* π^ρ is constructed by modifying the code of each ITM in π so that the first message sent to each copy of $\mathcal{F}$ is replaced with an invocation of a new copy of ρ with fresh random coins, with the same SID, and with the contents of that message as input. Each subsequent message to that copy of $\mathcal{F}$ is replaced with an activation of the corresponding copy of ρ, with the contents of that message given to ρ as new input. Each input value generated by a copy of ρ is treated as a message received from the corresponding copy of $\mathcal{F}$ (see [Can01] for more details). If ρ is a protocol in the real-life model then so is π^ρ. If ρ is a protocol in some $\mathcal{G}$-hybrid model, then so is π^ρ.

The composition theorem. In its general form, the composition theorem essentially states that if ρ UC realizes $\mathcal{F}$ in the $\mathcal{G}$-hybrid model for some functionality $\mathcal{G}$, then an execution of the composed protocol π^ρ, running in the $\mathcal{G}$-hybrid model, "emulates" an execution of π in the $\mathcal{F}$-hybrid model. That is, for any adversary $\mathcal{A}$ in the $\mathcal{G}$-hybrid model there exists an adversary $\mathcal{S}$ in the $\mathcal{F}$-hybrid model such that no environment machine $\mathcal{Z}$ can tell with non-negligible probability whether it is interacting with $\mathcal{A}$ and π^ρ in the $\mathcal{G}$-hybrid model or it is interacting with $\mathcal{S}$ and π in the $\mathcal{F}$-hybrid model. A corollary of the general theorem states that if π UC-realizes some functionality $\mathcal{I}$ in the $\mathcal{F}$-hybrid model, and ρ UC-realizes $\mathcal{F}$ in the $\mathcal{G}$-hybrid model, then π^ρ UC realizes $\mathcal{I}$ in the $\mathcal{G}$-hybrid model. Formally:

Theorem 4.1.2 ([Can01]). *Let $\mathcal{F}, \mathcal{G}, \mathcal{I}$ be ideal functionalities. Let π be a two-party* *protocol in the $\mathcal{F}$-hybrid model, and let ρ be a two-party protocol that UC realizes $\mathcal{F}$ in the $\mathcal{G}$-hybrid model. Then for any adversary $\mathcal{A}$ in the $\mathcal{G}$-hybrid model, there exists an adversary $\mathcal{S}$ in the $\mathcal{F}$-hybrid model such that for any environment machine $\mathcal{Z}$, we have:*

$$EXEC^{\mathcal{G}}_{\pi^{\rho},\mathcal{A},\mathcal{Z}} \overset{c}{\approx} EXEC^{\mathcal{F}}_{\pi,\mathcal{S},\mathcal{Z}}.$$

In particular, if π UC realizes functionality $\mathcal{I}$ in the $\mathcal{F}$-hybrid model then π^{ρ} UC realizes $\mathcal{I}$ in the $\mathcal{G}$-hybrid model.

4.1.2.4 Non-reactive functionalities

Our work deals with *non-reactive* functionalities. That is, functionalities that hold until they receive an input from each of the participating parties (in any order), then compute outputs (for any of the parties) and halt. We model a non-reactive functionality $\mathcal{F}$ with a family of circuits $\{F_k\}_{k\in\mathbb{N}}$. As a convention, we assume that $\mathcal{F}$ expects an input message with an action-description header field $\mathcal{F}\text{-}\mathsf{input}_i$ from party P_i, and produces an output message with an action-description header field $\mathcal{F}\text{-}\mathsf{output}_i$ for party P_i.

4.1.3 Universally Composable Zero Knowledge

Here, we overview the *ideal zero-knowledge* functionality $\mathcal{F}_{\mathrm{ZK}}$, following the treatment of [CLOS02], and discuss a non-interactive protocol that UC-realizes it in the presence of static adversaries. Looking ahead, our round-efficient, two-party computation constructions will be presented in the $\mathcal{F}_{\mathrm{ZK}}$-hybrid model.

In the zero-knowledge functionality, parameterized by a relation R, a *prover* sends the functionality a statement x to be proven, along with a witness w. In response, the functionality forwards the statement x to a *verifier* if and only if $R(x, w) = 1$ (i.e., if and only if it is a correct statement). The functionality is presented in Figure 4.1.

Functionality $\mathcal{F}_{\mathbf{ZK}}$

$\mathcal{F}_{\text{ZK}}$ proceeds as follows, running with a prover P, a verifier V and an adversary S, and parameterized with a relation R:

- Upon receiving $(\mathsf{ZK\text{-}prover}, sid, x, w)$ from P, do: if $R(x, w) = 1$, then send $(\mathsf{ZK\text{-}proof}, sid, x)$ to V and S and halt. Otherwise, halt.

Figure 4.1: The ideal Universally-Composable zero-knowledge functionality.

We note the following about the above formulation. First, in actuality, we have a proof of knowledge here, in that the verifier is assured that the prover *knows* w (and has explicitly sent w to the functionality), rather than just assured that such a w exists. Second, the functionality is defined so that only correct statements (i.e., values x such that $R(x, w) = 1$) are received by the verifier; incorrect statements are ignored by the functionality, and the verifier receives no notification that an attempt at cheating in the proof took place. This convention simplifies the description and analysis of our protocols. We note however that it is not essential; error messages can be added to the functionality (an realized) in a straightforward manner.

For the case of static adversaries, De Santis et al. [DDO+01] provide a non-interactive protocol (i.e., consists of a single message from the prover to the verifier) that UC realizes $\mathcal{F}_{\text{ZK}}$ for any NP relation (see also a discussion in [CLOS02, Section 6]); the protocol is given in the CRS model and assumes the existence of enhanced trapdoor-permutations (see [Gol04, Appendix C.1] for a discussion of this assumption).

4.1.4 Yao's "Garbled Circuit" Technique

Our protocol uses as a building block the "garbled-circuit" technique of Yao [Yao86]. We follow [KO04] in abstracting the technique, and consider those aspects of it which are necessary for our proofs of security; for the full construction and proof of security (when the participating parties are assumed to be honest-but-curious, and in the stand-alone setting), see [LP04].

Let F_k be a description of a two-input/single-output circuit whose inputs and output are of length k (the technique easily extends to lengths polynomial in k). Yao's results provide two PPT algorithms:

1. Yao_1 is a randomized algorithm which takes as input a security parameter $k \in \mathbb{N}$, a circuit F_k, and a string $y \in \{0,1\}^k$. It outputs a *garbled circuit* $\mathsf{Circuit}$ and *input-wire labels* $\{Z_{i,\sigma}\}_{i\in\{1,\ldots,k\},\sigma\in\{0,1\}}$.

2. Yao_2 is a deterministic algorithm which takes as input a security parameter $k \in \mathbb{N}$, a "garbled-circuit" $\mathsf{Circuit}$ and values $\{Z_i\}_{i\in\{1,\ldots,k\}}$ where $Z_i \in \{0,1\}^k$.

It outputs either an invalid symbol $\perp$, or a value $v \in \{0,1\}^k$.

We informally describe how the above algorithms may be used for secure computation when the participating parties are honest-but-curious. Let P_1 hold input $x = x_1 \dots x_k \in \{0,1\}^k$, P_2 hold input $y \in \{0,1\}^k$, and assume P_1 is to obtain the output $F_k(x,y)$. First, P_2 computes $(\mathsf{Circuit}, \{Z_{i,\sigma}\}_{i,\sigma}) \overset{R}{\leftarrow} \mathsf{Yao}_1(k, F_k, y)$ and sends $\mathsf{Circuit}$ to P_1. Then the players engage in k instances of *Oblivious Transfer*: in the i^{th} instance, P_1 enters with input x_i, P_2 enters with input $(Z_{i,0}, Z_{i,1})$, and P_1 obtains $Z_i \overset{\text{def}}{=} Z_{i,x_i}$ (additionally, P_2 "learns nothing" about x_i, and P_1 "learns nothing" about $Z_{i,1-x_i}$). P_1 then computes $v \leftarrow \mathsf{Yao}_2(\mathsf{Circuit}, \{Z_i\}_i)$, and outputs v.

With the above in mind, we describe the properties required of $\mathsf{Yao}_1, \mathsf{Yao}_2$. We first require *correctness*: for any F_k, y, any output $(\mathsf{Circuit}, \{Z_{i,\sigma}\}_{i,\sigma})$ of $\mathsf{Yao}_1(k, F_k, y)$ and any x, we have $F_k(x,y) = \mathsf{Yao}_2(k, \mathsf{Circuit}, \{Z_{i,x_i}\}_i)$. The algorithms also satisfy the following notion of *security*: there exists a PPT *simulator* $\mathsf{Yao\text{-}Sim}$ which takes k, F_k, x, v as inputs, and outputs $\mathsf{Circuit}$ and a set of k input-wire labels $\{Z_i\}_i$; furthermore, for any PPT A, the following two ensembles are computationally indistinguishable:

$$(1) \left\{ \begin{array}{c} (\mathsf{Circuit}, \{Z_{i,\sigma}\}_{i,\sigma}) \overset{R}{\leftarrow} \mathsf{Yao}_1(k, F_k, y) : \\ A(k, z, x, y, \mathsf{Circuit}, \{Z_{i,x_i}\}_i) \end{array} \right\}_{k \in \mathbb{N}, z \in \{0,1\}^*, x, y \in \{0,1\}^k}$$

$$(2) \left\{ \begin{array}{c} v = F_k(x,y) : \\ A(k, z, x, y, \mathsf{Yao\text{-}Sim}(k, F_k, x, v)) \end{array} \right\}_{k \in \mathbb{N}, z \in \{0,1\}^*, x, y \in \{0,1\}^k}.$$

4.1.5 The Decisional Diffie-Hellman (DDH) Assumption

We use a two-round *oblivious transfer (OT)* protocol as a building block in our constructions; any OT protocol based on *smooth projective hashing for hard subset-membership problems* per Tauman's framework [Tau05] will do. To simplify the exposition, we describe our constructions in terms of a protocol based on the Decisional Diffie-Hellman (DDH) assumption [DH76] which we recall here; we list known, alternative assumptions on which suitable OT protocols may be based in Section 4.2.

A *group generator* $\mathsf{GroupGen}$ is a PPT which on input $k \in \mathbb{N}$ outputs a description of a cyclic group $\mathcal{G}$ of prime order q, the order q and a generator $g \in \mathcal{G}$. Looking ahead, we will want to associate messages of length k with group elements; for simplicity we thus assume that $|q| \geq k$ (alternatively, we could use hashing).

Definition 4.1.3. We say the *Decisional Diffie-Hellman (DDH) problem is hard for* $\mathsf{GroupGen}$ if for any PPT algorithm A, the following ensembles are computationally indistinguishable:

$$(1)\ \left\{(\mathcal{G}, q, g) \overset{R}{\leftarrow} \mathsf{GroupGen}(k); a, b \overset{R}{\leftarrow} \mathbb{Z}_q : A(k, z, \mathcal{G}, q, g, g^a, g^b, g^{ab})\right\}_{k\in\mathbb{N}, z\in\{0,1\}^*}$$

$$(2)\ \left\{(\mathcal{G}, q, g) \overset{R}{\leftarrow} \mathsf{GroupGen}(k); a, b, c \overset{R}{\leftarrow} \mathbb{Z}_q : A(k, z, \mathcal{G}, q, g, g^a, g^b, g^{c})\right\}_{k\in\mathbb{N}, z\in\{0,1\}^*}. \diamondsuit$$

The following claim follows from the above definition via a simple hybrids argument.

Claim 4.1.4. *If the DDH problem is hard for* $\mathsf{GroupGen}$*, then for any PPT algorithm* A*, ensemble (1) above is also computationally indistinguishable from:*

$$(2')\ \left\{(\mathcal{G}, q, g) \overset{R}{\leftarrow} \mathsf{GroupGen}(k); a, b \overset{R}{\leftarrow} \mathbb{Z}_q : A(k, z, \mathcal{G}, q, g, g^a, g^b, g^{ab-1})\right\}_{k\in\mathbb{N}, z\in\{0,1\}^*}.$$

4.2 Round-Optimal, Universally Composable Two-Party Computation

In this section, we show a two-round protocol for Universally-Composable (UC) two-party computation. We reach our construction in two steps:

- First, we describe a two-round protocol, where parties take turns in speaking, for securely computing functionalities that provide output to only *one* of the participating parties. Using standard techniques, our protocol may be compiled into one that securely computes functionalities providing output to *both* parties at the cost of an additional communication round. The protocol is of interest in its own right (in Section 4.3, we use it to obtain a two-round UC blind-signature scheme, matching a recent result by Fischlin [Fis06]), but also serves as a stepping stone towards our main construction.
- We show how to bind and run two instances of our first protocol "in parallel", once in each "direction", so as to obtain a two-round protocol, where both parties speak in each round, for securely computing functionalities that provide output to *both* participating parties. We stress that our security analysis takes into account adversaries who may, in particular, wait to receive a message from their partner before sending their own in any given round (so called "rushing" adversaries).

At the end of the section, we observe that two rounds of communication are necessary for UC two-party computation when parties may speak simultaneously in any given round; and that three rounds are necessary for the task when parties need to take turns in sending their protocol messages.

Our constructions use UC zero-knowledge, Yao's garbled circuit technique and two-message oblivious transfer (OT) as building blocks. As mentioned earlier, any OT protocol based on *smooth projective hashing for hard subset-membership problems* per Tauman's framework [Tau05] will do. We stress that such OT protocols satisfy a weaker notion of security than the one needed here; we use zero-knowledge to lift the security guarantees to the level we need. To simplify the description of our protocols, we use a protocol from the framework based on the DDH assumption, simplifying a construction due to Naor and Pinkas [NP01]. We remark that other protocols conforming to Tauman's framework are known to exist under the DDH assumption [AIR01], under the N$^{\text{th}}$-residuosity assumption and under both the Quadratic-Residuosity assumption and the Extended Riemann hypothesis [Tau05].

4.2.1 A Two-Round Protocol for Single-Output Functionalities

Let $\mathcal{F} = \{F_k\}_{k \in \mathbb{N}}$ be a non-reactive, polynomial-sized, two-party functionality that provides output to a single party, say P_1. To simplify matters, we assume that $\mathcal{F}$ is deterministic; randomized functionalities can be handled using standard tools [Gol04, Prop. 7.4.4]. Without loss of generality, assume that F_k takes two k-bit inputs and produces a k-bit output (the protocol easily extends to input/output lengths polynomial in k). Let $\mathsf{GroupGen}$ be a group generator as in Section 4.1.5; recall our simplifying assumption that the order q of the group generated by $\mathsf{GroupGen}$

on security parameter k is such that $|q| \geq k$.

In this section, we describe a two round protocol for computing such $\mathcal{F}$ in the UC setting. Informally, our protocol proceeds as follows. Recall that in oblivious transfer, a receiver chooses and obtains one of two strings held by a sender, such that the sender "learns nothing" about the receiver's choice (preserving the receiver's privacy) while the receiver "learns nothing" about the second string (preserving the sender's privacy). The first round of our protocol is used to set up k instances of oblivious transfer. The second round is used to communicate a "garbled circuit" per Yao's construction, and for completing the oblivious-transfer of circuit input-wire labels that correspond to P_1's input (cf. Section 4.1.4).

To gain more intuition, we sketch a single oblivious transfer instance, assuming both parties are honest (the actual construction accounts for possibly malicious behavior by the parties with the aid of zero-knowledge). Let $\mathcal{G}$ be a group and g a generator, provided by GroupGen as above. To obtain the label corresponding to an input x_i for wire i, P_1 picks elements a, b uniformly at random from $\mathcal{G}$ and sends P_2 a tuple $(u = g^a, v = g^b, w = g^c)$, where c is set to ab if $x_i = 0$, to $(ab - 1)$ otherwise. Note that if the DDH problem is hard for GroupGen, P_2 will not be able to tell a tuple generated for $x_i = 0$ from one generated for $x_i = 1$, preserving P_1's privacy. Let $Z_{i,\sigma}$ be the label corresponding to input bit σ for wire i. P_2 selects r_0, s_0, r_1, s_1 uniformly at random from $\mathcal{G}$, and sends P_1 two pairs as follows:

$$(K_0 = u^{r_0} \cdot g^{s_0} \ , \ C_0 = w^{r_0} \cdot v^{s_0} \cdot Z_{i,0}) \text{ ; and}$$

$$(K_1 = u^{r_1} \cdot g^{s_1} \ , \ C_1 = (g \cdot w)^{r_1} \cdot v^{s_1} \cdot Z_{i,1}).$$

It is easy to verify that P_1 can obtain Z_{i,x_i} by computing $K_{x_i}^{-b} \cdot C_{x_i}$. Moreover, it can be shown that the tuple (K_{1-x_i}, C_{1-x_i}) is uniformly distributed (over the choice of r_{1-x_i}, s_{1-x_i}), and therefore P_1 "learns nothing" (information-theoretically) about the label corresponding to input $(1 - x_i)$ for wire i, preserving P_2's privacy.

In the following, we describe our two-round protocol $\pi_{\mathcal{F}}$ for UC realizing $\mathcal{F}$ in the $\mathcal{F}_{\mathrm{ZK}}$-hybrid model. In our description, we always let i range from 1 to k and σ range from 0 to 1.

Common Reference String: On security parameter $k \in \mathbb{N}$, the CRS is $(\mathcal{G}, q, g) \overset{R}{\leftarrow} \mathsf{GroupGen}(k)$.

First Round: P_1 on inputs $k \in \mathbb{N}$, $x = x_1 \ldots x_k \in \{0,1\}^k$ and sid, proceeds as follows:

1. For every i, chooses a_i, b_i uniformly at random from $\mathbb{Z}_q$, sets:

$$c_i = \begin{cases} a_i b_i & x_i = 0 \\ a_i b_i - 1 & \text{otherwise,} \end{cases}$$

   and lets $u_i = g^{a_i}, v_i = g^{b_i}, w_i = g^{c_i}$.

2. P_1 sends

$$(\mathsf{ZK\text{-}prover}, sid \circ 1, (\{u_i, v_i, w_i\}_i, (\mathcal{G}, q, g), k), (x, \{a_i, b_i\}_i))$$

   to $\mathcal{F}_{\mathrm{ZK}}^1$, where $\mathcal{F}_{\mathrm{ZK}}^1$ is parameterized by the relation:

$$R_1 = \left\{ ((\{u_i, v_i, w_i\}_i, (\mathcal{G}, q, g), k), (x, \{a_i, b_i\}_i)) \;\middle|\; \begin{array}{l} \forall i, u_i = g^{a_i}, v_i = g^{b_i}, w_i = g^{c_i}, \\ \text{where } c_i = \begin{cases} a_i b_i & x_i = 0 \\ a_i b_i - 1 & \text{otherwise} \end{cases} \end{array} \right\}$$

   and is set up such that P_1 is the prover and P_2 is the verifier.

Second Round: P_2, on inputs $k \in \mathbb{N}$, $y = y_1 \dots y_k \in \{0,1\}^k$ and sid, and upon receiving

$$(\textsf{ZK-proof}, sid \circ 1, (\{u_i, v_i, w_i\}_i, (\mathcal{G}', q', g'), k'))$$

from $\mathcal{F}^1_{\text{ZK}}$, first verifies that $\mathcal{G}' = \mathcal{G}, q' = q, g' = g$ and $k' = k$. If any of these conditions fail, P_2 ignores the message. Otherwise, it proceeds as follows:

1. Generates a "garbled circuit" (cf. Section 4.1.4) for F_k, based on its own input y. This involves choosing random coins Ω and computing $(\textsf{Circuit}, \{Z_{i,\sigma}\}_{i,\sigma}) \leftarrow \textsf{Yao}_1(k, F_k, y; \Omega)$.

2. For every i and σ, chooses $r_{i,\sigma}, s_{i,\sigma}$ uniformly at random from $\mathbb{Z}_q$, and sets:

$$K_{i,0} = u_i^{r_{i,0}} \cdot g^{s_{i,0}}, \quad C_{i,0} = w_i^{r_{i,0}} \cdot v_i^{s_{i,0}} \cdot Z_{i,0};$$
$$K_{i,1} = u_i^{r_{i,1}} \cdot g^{s_{i,1}}, \quad C_{i,1} = (g \cdot w_i)^{r_{i,1}} \cdot v_i^{s_{i,1}} \cdot Z_{i,1}.$$

3. Sends
$$\left(\textsf{ZK-prover}, sid \circ 2, \begin{pmatrix} \textsf{Circuit}, \{K_{i,\sigma}, C_{i,\sigma}\}_{i,\sigma} \\ (\mathcal{G}, q, g), k, \{u_i, v_i, w_i\}_i \end{pmatrix}, \begin{pmatrix} y, \Omega, \{Z_{i,\sigma}\}_{i,\sigma} \\ \{r_{i,\sigma}, s_{i,\sigma}\}_{i,\sigma} \end{pmatrix}\right)$$
to $\mathcal{F}^2_{\text{ZK}}$, where $\mathcal{F}^2_{\text{ZK}}$ is parameterized by the relation:
$$R_2 = \left\{ \left(\begin{matrix} \textsf{Circuit} \\ \{K_{i,\sigma}, C_{i,\sigma}\}_{i,\sigma} \\ (\mathcal{G}, q, g), k \\ \{u_i, v_i, w_i\}_i \end{matrix}, \begin{matrix} y, \Omega \\ \{Z_{i,\sigma}\}_{i,\sigma} \\ \{r_{i,\sigma}, s_{i,\sigma}\}_{i,\sigma} \end{matrix} \right) \middle| \begin{matrix} (\textsf{Circuit}, \{Z_{i,\sigma}\}_{i,\sigma}) = \textsf{Yao}_1(k, F_k, y; \Omega) \\ \wedge \forall i, \\ K_{i,0} = u_i^{r_{i,0}} \cdot g^{s_{i,0}}, C_{i,0} = w_i^{r_{i,0}} \cdot v_i^{s_{i,0}} \cdot Z_{i,0}; \\ K_{i,1} = u_i^{r_{i,1}} \cdot g^{s_{i,1}}, C_{i,1} = (g \cdot w_i)^{r_{i,1}} \cdot v_i^{s_{i,1}} \cdot Z_{i,1} \end{matrix} \right\}$$
and is set up such that P_2 is the prover and P_1 is the verifier.

Output Computation: P_1, upon receipt of message

$$(\textsf{ZK-proof}, sid \circ 2, (\textsf{Circuit}, \{K_{i,\sigma}, C_{i,\sigma}\}_{i,\sigma}, (\mathcal{G}', q', g'), k', \{u_i', v_i', w_i'\}_i))$$

from $\mathcal{F}^2_{\mathrm{ZK}}$, first verifies that $\mathcal{G}' = \mathcal{G}, q' = q, g' = g, k' = k$ and $\{u'_i, v'_i, w'_i\}_i = \{u_i, v_i, w_i\}_i$. If any of these conditions fail, P_1 ignores the message. Otherwise, it completes the protocol by computing $Z_i \overset{\text{def}}{=} K_{i,x_i}^{-b_i} \cdot C_{i,x_i}$, computing $v \leftarrow \mathsf{Yao}_2(k, \mathsf{Circuit}, \{Z_i\}_i)$ and reporting v as output if $v \neq \perp$.

Concrete round complexity. When composed with the non-interactive protocol of De Santis et al. [DDO+01] UC-realizing F_{ZK}, our protocol takes two communication rounds. Its security now additionally rests on the existence of enhanced trapdoor permutations.

Security. We forgo the analysis here — the protocol may be viewed as a degenerate version of the construction we present and analyze next.

Compiling the protocol into one computing two-output functionalities. The protocol may be compiled into one that securely computes functionalities providing output to both parties at the cost of an additional round of communication using standard techniques [Gol04, Prop. 7.2.11].

4.2.2 A Two-Round Protocol for Two-Output Functionalities

Let $\mathcal{F} = \left\{F_k \overset{\text{def}}{=} (F_k^1, F_k^2)\right\}_{k \in \mathbb{N}}$ be a non-reactive, polynomial-sized, two-party functionality such that P_1 wishes to obtain $F_k^1(x, y)$ and P_2 wishes to obtain $F_k^2(x, y)$ when P_1 holds x and P_2 holds y. Without loss of generality, assume once more that $\mathcal{F}$ is deterministic; that x, y and the outputs of F_k^1, F_k^2 are k-bit strings; and that the order q of a group generated by $\mathsf{GroupGen}$ on security parameter k is such that $|q| \geq k$; see Section 4.2.1.

The protocol of the preceding section provides means to securely compute a functionality that provides output to *one* of the parties, in two rounds. To securely-compute our two-output functionality $F_k = (F_k^1, F_k^2)$, we run one instance of that protocol such that P_1 receives F_k^1 (with a first-round message originating from P_1 and a second-round message from P_2), and a second instance such that P_2 receives F_k^2 (with a first-round message originating from P_2 and a second-round message from P_1); if we allow the parties to transmit messages *simultaneously* in any given round, this yields a two-round protocol. All that's left to ensure is that each party enters both instances of the protocol with the same input; we have the relation parameterizing the second-round zero-knowledge functionality enforce this condition[1].

Below, we describe our two-round protocol $\pi_{\mathcal{F}}$ for UC realizing $\mathcal{F}$ in the $\mathcal{F}_{\text{ZK}}$-hybrid model when parties are allowed to send messages simultaneously in any given round. We describe our protocol from the perspective of P_1; P_2 behaves analogously (i.e., the protocol is symmetric). In the description, we always let i range from 1 to k and σ range from 0 to 1.

Common Reference String: On security parameter $k \in \mathbb{N}$, the CRS is $(\mathcal{G}, q, g) \overset{R}{\leftarrow} \mathsf{GroupGen}(k)$.

First Round: P_1 on inputs $k \in \mathbb{N}$, $x = x_1 \dots x_k \in \{0,1\}^k$ and *sid*, proceeds as

[1]Alternatively, we can make the following modifications to the protocol of the preceding section: each party will add a commitment to its input to its original protocol message, and modify its zero-knowledge assertion to reflect that it has constructed the original message with an input that is consistent with the commitment. Two instances of this protocol can now be run in parallel as above without further modifications (note that the second-round commitments become redundant).

follows:

1. For every i, chooses a_i, b_i uniformly at random from $\mathbb{Z}_q$, sets:

$$c_i = \begin{cases} a_i b_i & x_i = 0 \\ a_i b_i - 1 & \text{otherwise,} \end{cases}$$

   and lets $u_i = g^{a_i}, v_i = g^{b_i}, w_i = g^{c_i}$.

2. Sends

$$(\mathsf{ZK\text{-}prover}, sid \circ 1 \circ P_1, (\{u_i, v_i, w_i\}_i, (\mathcal{G}, q, g), k), (x, \{a_i, b_i\}_i))$$

   to $\mathcal{F}_{\mathrm{ZK}}^{1,P_1 \to P_2}$, where $\mathcal{F}_{\mathrm{ZK}}^{1,P_1 \to P_2}$ is parameterized by the relation:

$$R_1 = \left\{ ((\{u_i, v_i, w_i\}_i, (\mathcal{G}, q, g), k), (x, \{a_i, b_i\}_i)) \;\middle|\; \begin{array}{l} \forall i, u_i = g^{a_i}, v_i = g^{b_i}, w_i = g^{c_i}, \\ \text{where } c_i = \begin{cases} a_i b_i & x_i = 0 \\ a_i b_i - 1 & \text{otherwise} \end{cases} \end{array} \right\}$$

   and is set up such that P_1 is the prover and P_2 is the verifier.

Second Round: Upon receiving the symmetric first-round message

$$(\mathsf{ZK\text{-}proof}, sid \circ 1 \circ P_2, (\{\bar{u}_i, \bar{v}_i, \bar{w}_i\}_i, (\mathcal{G}', q', g'), k'))$$

from $\mathcal{F}_{\mathrm{ZK}}^{1,P_2 \to P_1}$ (defined analogously to $\mathcal{F}_{\mathrm{ZK}}^{1,P_1 \to P_2}$ using the relation R_1, but set up such that P_2 is the prover and P_1 is the verifier), P_1 verifies that $\mathcal{G}' = \mathcal{G}, q' = q, g' = g$ and $k' = k$. If any of these conditions fail, P_1 ignores the message. Otherwise, it proceeds as follows:

1. Generates a "garbled circuit" (cf. Section 4.1.4) for F_k^2, based on its own input x. This involves choosing random coins Ω and computing $(\mathsf{Circuit}, \{Z_{i,\sigma}\}_{i,\sigma}) \leftarrow \mathsf{Yao}_1(k, F_k^2, x; \Omega)$.

2. For every i and σ, chooses $r_{i,\sigma}, s_{i,\sigma}$ uniformly at random from $\mathbb{Z}_q$, and sets:

$$K_{i,0} = \bar{u}_i^{r_{i,0}} \cdot g^{s_{i,0}}, \quad C_{i,0} = \bar{w}_i^{r_{i,0}} \cdot \bar{v}_i^{s_{i,0}} \cdot Z_{i,0};$$
$$K_{i,1} = \bar{u}_i^{r_{i,1}} \cdot g^{s_{i,1}}, \quad C_{i,1} = (g \cdot \bar{w}_i)^{r_{i,1}} \cdot \bar{v}_i^{s_{i,1}} \cdot Z_{i,1}.$$

3. Sends

$$\left(\mathsf{ZK\text{-}prover}, sid \circ 2 \circ P_1, \begin{pmatrix} \mathsf{Circuit}, \{K_{i,\sigma}, C_{i,\sigma}\}_{i,\sigma} \\ (\mathcal{G}, q, g), k, \{\bar{u}_i, \bar{v}_i, \bar{w}_i\}_i \\ \{u_i, v_i, w_i\}_i \end{pmatrix}, \begin{pmatrix} x, \Omega, \{Z_{i,\sigma}\}_{i,\sigma} \\ \{r_{i,\sigma}, s_{i,\sigma}\}_{i,\sigma} \\ \{a_i, b_i\}_i \end{pmatrix}\right)$$

to $\mathcal{F}_{\mathrm{ZK}}^{2,P_1 \to P_2}$, where $\mathcal{F}_{\mathrm{ZK}}^{2,P_1 \to P_2}$ is parameterized by the relation:

$$R_2 = \left\{ \left(\begin{array}{c} \mathsf{Circuit} \\ \{K_{i,\sigma}, C_{i,\sigma}\}_{i,\sigma} \\ (\mathcal{G}, q, g), k \\ \{\bar{u}_i, \bar{v}_i, \bar{w}_i\}_i \\ \{u_i, v_i, w_i\}_i \end{array}, \begin{array}{c} x, \Omega \\ \{Z_{i,\sigma}\}_{i,\sigma} \\ \{r_{i,\sigma}, s_{i,\sigma}\}_{i,\sigma} \\ \{a_i, b_i\}_i \end{array} \right) \middle| \begin{array}{c} (\mathsf{Circuit}, \{Z_{i,\sigma}\}_{i,\sigma}) = \mathsf{Yao}_1(k, F_k^2, x; \Omega) \\ \wedge\ \forall i, \\ K_{i,0} = \bar{u}_i^{r_{i,0}} \cdot g^{s_{i,0}},\ C_{i,0} = \bar{w}_i^{r_{i,0}} \cdot \bar{v}_i^{s_{i,0}} \cdot Z_{i,0} \\ K_{i,1} = \bar{u}_i^{r_{i,1}} \cdot g^{s_{i,1}},\ C_{i,1} = (g \cdot \bar{w}_i)^{r_{i,1}} \cdot \bar{v}_i^{s_{i,1}} \cdot Z_{i,1} \\ \wedge\ \forall i,\ u_i = g^{a_i}, v_i = g^{b_i}, w_i = g^{c_i}, \\ \text{where } c_i = \begin{cases} a_i b_i & x_i = 0 \\ a_i b_i - 1 & \text{otherwise} \end{cases} \end{array} \right\}$$

and is set up such that P_1 is the prover and P_2 is the verifier.

Output Computation: Upon receiving the symmetric second-round message

$$(\mathsf{ZK\text{-}proof}, sid \circ 2 \circ P_2, (\overline{\mathsf{Circuit}}, \{\bar{K}_{i,\sigma}, \bar{C}_{i,\sigma}\}_{i,\sigma}, (\mathcal{G}', q', g'), k', \{u_i', v_i', w_i'\}_i, \{\bar{u}_i', \bar{v}_i', \bar{w}_i'\}_i))$$

from $\mathcal{F}_{\text{ZK}}^{2,P_2 \to P_1}$ (defined analogously to $\mathcal{F}_{\text{ZK}}^{2,P_1 \to P_2}$ using the relation R_2, but set up such that P_2 is the prover and P_1 is the verifier), P_1 verifies that $\mathcal{G}' = \mathcal{G}, q' = q, g' = g, k' = k$, that $\{u'_i, v'_i, w'_i\}_i = \{u_i, v_i, w_i\}_i$ and that $\{\bar{u}'_i, \bar{v}'_i, \bar{w}'_i\}_i = \{\bar{u}_i, \bar{v}_i, \bar{w}_i\}_i$. If any of these conditions fail, P_1 ignores the message. Otherwise, it completes the protocol by computing $\bar{Z}_i \overset{\text{def}}{=} \bar{K}_{i,x_i}^{-b_i} \cdot \bar{C}_{i,x_i}$, computing $v \leftarrow \mathsf{Yao}_2(k, \overline{\mathsf{Circuit}}, \{\bar{Z}_i\}_i)$ and reporting v as output if $v \neq \perp$.

Concrete round complexity. When composed with the non-interactive protocol of De Santis et al. [DDO$^+$01] realizing $\mathcal{F}_{\text{ZK}}$, the protocol takes two rounds of communication; its security now additionally requires the existence of enhanced trapdoor permutations.

Theorem 4.2.1. *Assuming that the DDH problem is hard for* $\mathsf{GroupGen}$*, the above protocol UC-realizes* $\mathcal{F}$ *in the* $\mathcal{F}_{\text{ZK}}$*-hybrid model (in the presence of static adversaries).*

Let $\mathcal{A}$ be a (static) adversary operating against $\pi_{\mathcal{F}}$ in the $\mathcal{F}_{\text{ZK}}$-hybrid model. To prove the theorem, we construct a simulator $\mathcal{S}$ such that no environment $\mathcal{Z}$ can tell with a non-negligible probability whether it is interacting with $\mathcal{A}$ and P_1, P_2 running $\pi_{\mathcal{F}}$ in the $\mathcal{F}_{\text{ZK}}$-hybrid model or with $\mathcal{S}$ and $\tilde{P}_1, \tilde{P}_2$ in the ideal process for $\mathcal{F}$. $\mathcal{S}$ will internally run a copy of $\mathcal{A}$, "simulating" for it an execution of $\pi_{\mathcal{F}}$ in the $\mathcal{F}_{\text{ZK}}$-hybrid model (by simulating an environment, a CRS, ideal $\mathcal{F}_{\text{ZK}}$ functionalities and parties P_1, P_2) that matches $\mathcal{S}$'s view of the ideal process; $\mathcal{S}$ will use $\mathcal{A}$'s actions to guide its own in the ideal process. We refer to an event as occurring in the *internal simulation* if it happens within the execution environment that $\mathcal{S}$ simulates for A.

We refer to an event as occurring in the *external process* if it happens within the ideal process, in which $\mathcal{S}$ is participating.

For clarity, we group $\mathcal{S}$'s actions according to the subset of parties that $\mathcal{A}$ has corrupted. Recall that $\mathcal{S}$ is given a security parameter $k \in \mathbb{N}$. $\mathcal{S}$ proceeds as follows:

Initial activation: $\mathcal{S}$ sets the (simulated) CRS to be $(\mathcal{G}, q, g) \overset{R}{\leftarrow} \mathsf{GroupGen}(k)$. It copies the input value written by $\mathcal{Z}$ on its own input tape onto $\mathcal{A}$'s input tape and activates $\mathcal{A}$. If $\mathcal{A}$ corrupts party P_i (in the internal simulation), $\mathcal{S}$ corrupts $\tilde{P}_i$ (in the external process). When $\mathcal{A}$ completes its activation, $\mathcal{S}$ copies the output value written by $\mathcal{A}$ on its output tape to $\mathcal{S}$'s own output tape, and ends its activation.

P_2 only is corrupted: Upon activation, $\mathcal{S}$ copies the input value written by $\mathcal{Z}$ on its own input tape onto $\mathcal{A}$'s input tape. In addition, if $\tilde{P}_1$ has added a message $(\mathcal{F}\text{-}\mathsf{input}_1, sid, \cdot)$ for $\mathcal{F}$ to its outgoing communication tape (in the external process; recall that $\mathcal{S}$ can only read the *public headers* of messages on the outgoing communication tapes of uncorrupted dummy parties), $\mathcal{S}$, for every i, chooses a_i, b_i uniformly at random from $\mathbb{Z}_q$, sets $u_i = g^{a_i}, v_i = g^{b_i}, w_i = g^{a_i b_i}$ for future use, and adds a message $(\mathsf{ZK\text{-}prover}, sid \circ 1 \circ P_1, \perp, \perp)$ for $\mathcal{F}_{\mathrm{ZK}}^{1, P_1 \to P_2}$ to P_1's outgoing communication tape (in the internal simulation; recall that A will only be able to read the *public header* of a message intended for $\mathcal{F}_{\mathrm{ZK}}$ on the outgoing communication tape of an uncorrupted party in the $\mathcal{F}_{\mathrm{ZK}}$-hybrid model). $\mathcal{S}$ then activates $\mathcal{A}$.

Upon completion of $\mathcal{A}$'s activation, $\mathcal{S}$ acts as follows:

1. If $\mathcal{A}$ delivered the message $(\textsf{ZK-prover}, sid \circ 1 \circ P_1, \perp, \perp)$ from P_1 to $\mathcal{F}_{\mathrm{ZK}}^{1,P_1 \to P_2}$ (in the internal simulation), $\mathcal{S}$ adds the message

$$(\textsf{ZK-proof}, sid \circ 1 \circ P_1, (\{u_i, v_i, w_i\}_i, (\mathcal{G}, q, g), k))$$

for P_2 and $\mathcal{A}$ to $\mathcal{F}_{\mathrm{ZK}}^{1,P_1 \to P_2}$'s outgoing communication tape (in the internal simulation). Informally, $\mathcal{S}$ constructs the message from $\mathcal{F}_{\mathrm{ZK}}^{1,P_1 \to P_2}$ to P_2 and $\mathcal{A}$ (in the internal simulation) in accordance with $\pi_{\mathcal{F}}$, except that it always lets w_i be $g^{a_i b_i}$.

2. If $\mathcal{A}$ delivered a message

$$(\textsf{ZK-prover}, sid \circ 1 \circ P_2, (\{\bar{u}_i, \bar{v}_i, \bar{w}_i\}_i, (\mathcal{G}', q', g'), k'), (y, \{\bar{a}_i, \bar{b}_i\}_i))$$

from P_2 to $\mathcal{F}_{\mathrm{ZK}}^{1,P_2 \to P_1}$ (in the internal simulation), $\mathcal{S}$ verifies that

$$((\{\bar{u}_i, \bar{v}_i, \bar{w}_i\}_i, (\mathcal{G}', q', g'), k'), (y, \{\bar{a}_i, \bar{b}_i\}_i)) \in R_1.$$

If the verification fails, $\mathcal{S}$ does nothing. Otherwise, $\mathcal{S}$ adds the message

$$(\textsf{ZK-proof}, sid \circ 1 \circ P_2, (\{\bar{u}_i, \bar{v}_i, \bar{w}_i\}_i, (\mathcal{G}', q', g'), k'))$$

for P_1 and $\mathcal{A}$ to $\mathcal{F}_{\mathrm{ZK}}^{1,P_2 \to P_1}$'s outgoing communication tape (in the internal simulation), and delivers the message $(\mathcal{F}\textsf{-input}_2, sid, y)$ from (the corrupted) $\tilde{P}_2$ to $\mathcal{F}$ (in the external simulation). $\mathcal{S}$ records the values y and $\{\bar{u}_i, \bar{v}_i, \bar{w}_i\}_i$.

3. If $\mathcal{A}$ delivered the message

$$(\textsf{ZK-proof}, sid \circ 1 \circ P_2, (\{\bar{u}_i, \bar{v}_i, \bar{w}_i\}_i, (\mathcal{G}', q', g'), k'))$$

from $\mathcal{F}_{\text{ZK}}^{1,P_2 \to P_1}$ to P_1 (in the internal simulation), $\mathcal{S}$ first verifies that $\tilde{P}_1$ has a message $(\mathcal{F}\text{-}\mathsf{input}_1, sid, \cdot)$ for $\mathcal{F}$ on its outgoing communication tape (in the external process) and that $\mathcal{G}' = \mathcal{G}, q' = q, g' = g$ and $k' = k$. If any of these fail, $\mathcal{S}$ does nothing. Otherwise, it adds the message $(\mathsf{ZK\text{-}prover}, sid \circ 2 \circ P_1, \bot, \bot)$ for $\mathcal{F}_{\text{ZK}}^{2,P_1 \to P_2}$ to P_1's outgoing communication tape (in the internal simulation), delivers $(\mathcal{F}\text{-}\mathsf{input}_1, sid, \cdot)$ from $\tilde{P}_1$ to $\mathcal{F}$ (in the external process), and notes to itself that the Round-1 message from $\mathcal{F}_{\text{ZK}}^{1,P_2 \to P_1}$ to P_1 (in the internal simulation) has been delivered. Note that once the activation of $\mathcal{S}$ will be complete, $\mathcal{F}$ will be in possession of both its inputs and will be activated next (in the external process).

4. If $\mathcal{A}$ delivered the message $(\mathsf{ZK\text{-}prover}, sid \circ 2 \circ P_1, \bot, \bot)$ from P_1 to $\mathcal{F}_{\text{ZK}}^{2,P_1 \to P_2}$, $\mathcal{S}$ proceeds as follows. First note that at this point, we are guaranteed that two inputs were delivered to $\mathcal{F}$ and that $\mathcal{F}$ has been activated subsequently (in the external process); therefore, $\mathcal{F}$ has written a message $(\mathcal{F}\text{-}\mathsf{output}_2, sid, v)$ for $\tilde{P}_2$ on its outgoing communication tape (in the external process; note that $\mathcal{S}$ may read the contents of a message from $\mathcal{F}$ to a corrupted party). Also note that at this point, $\mathcal{S}$ has recorded values y and $\{\bar{u}_i, \bar{v}_i, \bar{w}_i\}_i$ sent by (the corrupted) P_2 in its first-round message to $\mathcal{F}_{\text{ZK}}^{1,P_2 \to P_1}$. $\mathcal{S}$ produces a simulated "garbled circuit" and input-wire labels using F_k^2, y and v (cf. Section 4.1.4) by computing $(\mathsf{Circuit}, \{Z_i\}_i) \overset{R}{\leftarrow} \mathsf{Yao\text{-}Sim}(k, F_k^2, y, v)$. For every i, it chooses r_{i,y_i}, s_{i,y_i}

uniformly at random from $\mathbb{Z}_q$, sets:

$$K_{i,y_i} = \bar{u}_i^{r_{i,y_i}} \cdot g^{s_{i,y_i}}$$
$$C_{i,y_i} = \begin{cases} \bar{w}_i^{r_{i,y_i}} \cdot \bar{v}_i^{s_{i,y_i}} \cdot Z_i & \text{if } y_i = 0 \\ (g \cdot \bar{w}_i)^{r_{i,y_i}} \cdot \bar{v}_i^{s_{i,y_i}} \cdot Z_i & \text{otherwise,} \end{cases}$$

and sets $K_{i,1-y_i}, C_{i,1-y_i}$ to be elements selected uniformly at random from $\mathcal{G}$. It then adds the message

$$\begin{pmatrix} & \mathsf{Circuit}, \{K_{i,\sigma}, C_{i,\sigma}\}_{i,\sigma} \\ \mathsf{ZK\text{-}proof}, sid \circ 2 \circ P_1, & (\mathcal{G}, q, g), k, \{\bar{u}_i, \bar{v}_i, \bar{w}_i\}_i \\ & \{u_i, v_i, w_i\}_i \end{pmatrix}$$

for P_2 and $\mathcal{A}$ to the outgoing communication tape of $\mathcal{F}_{\text{ZK}}^{2,P_1 \to P_2}$. Informally, $\mathcal{S}$ constructs the message in accordance with $\pi_{\mathcal{F}}$, except that it uses simulated circuit and input wire labels, and sets $\{K_{i,1-y_i}, C_{i,1-y_i}\}_i$ to be uniform elements in $\mathcal{G}$.

5. If $\mathcal{A}$ delivered a message

$$\begin{pmatrix} & \overline{\mathsf{Circuit}}, \{\bar{K}_{i,\sigma}, \bar{C}_{i,\sigma}\}_{i,\sigma} & y', \bar{\Omega}, \{\bar{Z}_{i,\sigma}\}_{i,\sigma} \\ \mathsf{ZK\text{-}prover}, sid \circ 2 \circ P_2 \quad , & (\mathcal{G}', q', g'), k', \{u'_i, v'_i, w'_i\}_i \quad , & \{\bar{r}_{i,\sigma}, \bar{s}_{i,\sigma}\}_{i,\sigma} \\ & \{\bar{u}'_i, \bar{v}'_i, \bar{w}'_i\}_i & \{\bar{a}'_i, \bar{b}'_i\}_i \end{pmatrix}$$

from $\mathcal{A}$ to $\mathcal{F}_{\text{ZK}}^{2,P_2 \to P_1}$ (in the internal simulation), $\mathcal{S}$ verifies that

$$\begin{pmatrix} \overline{\mathsf{Circuit}}, \{\bar{K}_{i,\sigma}, \bar{C}_{i,\sigma}\}_{i,\sigma} & y', \bar{\Omega}, \{\bar{Z}_{i,\sigma}\}_{i,\sigma} \\ (\mathcal{G}', q', g'), k', \{u'_i, v'_i, w'_i\}_i \quad , & \{\bar{r}_{i,\sigma}, \bar{s}_{i,\sigma}\}_{i,\sigma} \\ \{\bar{u}'_i, \bar{v}'_i, \bar{w}'_i\}_i & \{\bar{a}'_i, \bar{b}'_i\}_i \end{pmatrix} \in R_2.$$

If the verification fails, $\mathcal{S}$ does nothing. Otherwise, $\mathcal{S}$ adds the message

$$\left(\begin{array}{cc} & \overline{\mathsf{Circuit}}, \left\{\bar{K}_{i,\sigma}, \bar{C}_{i,\sigma}\right\}_{i,\sigma} \\ \mathsf{ZK\text{-}proof}, sid \circ 2 \circ P_2, & (\mathcal{G}', q', g'), k', \{u'_i, v'_i, w'_i\}_i \\ & \{\bar{u}'_i, \bar{v}'_i, \bar{w}'_i\}_i \end{array}\right)$$

for P_1 and $\mathcal{A}$ to $\mathcal{F}_{\mathrm{ZK}}^{2,P_2\to P_1}$'s outgoing communication tape (in the internal simulation).

6. If $\mathcal{A}$ delivered the message

$$\left(\begin{array}{cc} & \overline{\mathsf{Circuit}}, \left\{\bar{K}_{i,\sigma}, \bar{C}_{i,\sigma}\right\}_{i,\sigma} \\ \mathsf{ZK\text{-}proof}, sid \circ 2 \circ P_2, & (\mathcal{G}', q', g'), k', \{u'_i, v'_i, w'_i\}_i \\ & \{\bar{u}'_i, \bar{v}'_i, \bar{w}'_i\}_i \end{array}\right)$$

from $\mathcal{F}_{\mathrm{ZK}}^{2,P_2\to P_1}$ to P_1 (in the internal simulation), $\mathcal{S}$ first checks whether a Round-1 message from $\mathcal{F}_{\mathrm{ZK}}^{1,P_2\to P_1}$ to P_1 (in the internal simulation) has been delivered, per Item 3 above; if not, $\mathcal{S}$ does nothing. Otherwise, we are guaranteed that two inputs were delivered to $\mathcal{F}$, and that $\mathcal{F}$ has subsequently been activated and written a message $(\mathcal{F}\text{-}\mathsf{output}_1, sid, \cdot)$ for $\tilde{P}_1$ on its outgoing communication tape (in the external process). $\mathcal{S}$ verifies that $\mathcal{G}' = \mathcal{G}, q' = q, g' = g, k' = k$, that $\{u'_i, v'_i, w'_i\}_i = \{u_i, v_i, w_i\}_i$ and that $\{\bar{u}'_i, \bar{v}'_i, \bar{w}'_i\}_i = \{\bar{u}_i, \bar{v}_i, \bar{w}_i\}_i$ (intuitively, the checks, along with those performed by $\mathcal{S}$ on behalf of $\mathcal{F}_{\mathrm{ZK}}^{2,P_2\to P_1}$ per Item 5 above, guarantee that (the corrupted) P_2 has used the same input consistently in both rounds, i.e., that $y' = y$); if so, $\mathcal{S}$ delivers the message $(\mathcal{F}\text{-}\mathsf{output}_1, sid, \cdot)$ from $\mathcal{F}$ to $\tilde{P}_1$ (in the external process).

After performing one of the above (if any), $\mathcal{S}$ copies the output value written by $\mathcal{A}$ on its output tape to $\mathcal{S}$'s own output tape, and ends its activation.

P_1 only is corrupted: Symmetric to the case where P_2 only is corrupted.

P_1, P_2 are both corrupted: Informally, $\mathcal{S}$ need only pass messages between $\mathcal{Z}$ and $\mathcal{A}$, and simulate $\mathcal{F}_{\text{ZK}}$ for $\mathcal{A}$. Specifically, upon activation, $\mathcal{S}$ copies the input value written by $\mathcal{Z}$ on its own input tape (in the external process) onto $\mathcal{A}$'s input tape and activates $\mathcal{A}$ (in the internal simulation). Upon the completion of $\mathcal{A}$'s activation, $\mathcal{S}$ proceeds as follows: if $\mathcal{A}$ delivered a message $(\mathsf{ZK\text{-}prover}, sid \circ \ell \circ P_i, \hat{x}, \hat{w})$ from P_i to $\mathcal{F}_{\text{ZK}}^{\ell, P_i \to P_j}$, $\mathcal{S}$ verifies that $(\hat{x}, \hat{w}) \in R_\ell$. If the verification fails, $\mathcal{S}$ does nothing; otherwise, it adds the message $(\mathsf{ZK\text{-}proof}, sid \circ \ell \circ P_i, \hat{x})$ for P_j and $\mathcal{A}$ to the outgoing communication tape of $\mathcal{F}_{\text{ZK}}^{\ell, P_i \to P_j}$. $\mathcal{S}$ copies the output value written by $\mathcal{A}$ on its output tape to $\mathcal{S}$'s own output tape, and ends its activation.

Neither P_1 nor P_2 is corrupted: Informally, $\mathcal{S}$ need only pass messages between $\mathcal{Z}$ and $\mathcal{A}$, and deliver messages in the external process in accordance to message delivery by $\mathcal{A}$ in the internal simulation. Specifically, upon activation, $\mathcal{S}$ copies the input value written by $\mathcal{Z}$ on its own input tape onto $\mathcal{A}$'s input tape. In addition, if $\tilde{P}_i$ has added a message $(\mathcal{F}\text{-}\mathsf{input}_i, sid, \cdot)$ for $\mathcal{F}$ to its outgoing communication tape (in the external process; recall that $\mathcal{S}$ can only read the *public headers* of messages on the outgoing communication tapes of uncorrupted dummy parties), $\mathcal{S}$ adds a message $(\mathsf{ZK\text{-}prover}, sid \circ 1 \circ P_i, \perp, \perp)$ for $\mathcal{F}_{\text{ZK}}^{1, P_i \to P_j}$ to P_i's outgoing communication tape (in the internal simulation;

recall that A will only be able to read the *public header* of a message intended for $\mathcal{F}_{\text{ZK}}$ on the outgoing communication tape of an uncorrupted party in the $\mathcal{F}_{\text{ZK}}$-hybrid model), where P_j is P_i's partner (i.e., $P_j = P_{(3-i)}$). $\mathcal{S}$ then activates $\mathcal{A}$.

Upon completion of $\mathcal{A}$'s activation, $\mathcal{S}$ acts as follows:

1. If $\mathcal{A}$ delivered the message $(\textsf{ZK-prover}, sid \circ 1 \circ P_i, \perp, \perp)$ from P_i to $\mathcal{F}_{\text{ZK}}^{1,P_i \to P_j}$ (in the internal simulation), $\mathcal{S}$ adds the message $(\textsf{ZK-proof}, sid \circ 1 \circ P_i, \perp)$ for P_j and $\mathcal{A}$ to the outgoing communication tape of $\mathcal{F}_{\text{ZK}}^{1,P_i \to P_j}$ (in the internal simulation).

2. If $\mathcal{A}$ delivered the message $(\textsf{ZK-proof}, sid \circ 1 \circ P_i, \perp)$ from $\mathcal{F}_{\text{ZK}}^{1,P_i \to P_j}$ to P_j (in the internal simulation), note first that $\mathcal{A}$ must have delivered the message $(\textsf{ZK-prover}, sid \circ 1 \circ P_i, \perp, \perp)$ from P_i to $\mathcal{F}_{\text{ZK}}^{1,P_i \to P_j}$ (in the internal simulation) per Item 1 above, and so it must be the case that $\tilde{P}_i$ has a message $(\mathcal{F}\textsf{-input}_i, sid, \cdot)$ for $\mathcal{F}$ on its outgoing communication tape (in the external process). $\mathcal{S}$ verifies that $\tilde{P}_j$ has added a message $(\mathcal{F}\textsf{-input}_j, sid, \cdot)$ for $\mathcal{F}$ to its outgoing communication tape (in the external process). If not, $\mathcal{S}$ does nothing. Otherwise, it adds the message $(\textsf{ZK-prover}, sid \circ 2 \circ P_j, \perp, \perp)$ for $\mathcal{F}_{\text{ZK}}^{2,P_j \to P_i}$ to the outgoing communication tape of P_j (in the internal simulation), delivers the message $(\mathcal{F}\textsf{-input}_i, sid, \cdot)$ from $\tilde{P}_i$ to $\mathcal{F}$ (in the external process), and notes to itself that the Round-1 message from $\mathcal{F}_{\text{ZK}}^{1,P_i \to P_j}$ to P_j (in the internal simulation) has been delivered.

3. If $\mathcal{A}$ delivered the message $(\textsf{ZK-prover}, sid \circ 2 \circ P_i, \perp, \perp)$ from P_i to $\mathcal{F}_{\text{ZK}}^{2,P_i \to P_j}$

(in the internal simulation), $\mathcal{S}$ adds the message $(\textsf{ZK-proof}, sid \circ 2 \circ P_i, \perp)$ for P_j and $\mathcal{A}$ to the outgoing communication tape of $\mathcal{F}_{\text{ZK}}^{2,P_i \to P_j}$ (in the internal simulation).

4. If $\mathcal{A}$ delivered the message $(\textsf{ZK-proof}, sid \circ 2 \circ P_i, \perp)$ from $\mathcal{F}_{\text{ZK}}^{2,P_i \to P_j}$ to P_j, note first that $\mathcal{A}$ must have delivered the message $(\textsf{ZK-prover}, sid \circ 2 \circ P_i, \perp, \perp)$ from P_i to $\mathcal{F}_{\text{ZK}}^{2,P_i \to P_j}$ (in the internal simulation) per Item 3 above, and so that the Round-1 message from $\mathcal{F}_{\text{ZK}}^{1,P_j \to P_i}$ to P_i (in the internal simulation) has been delivered as well, per Item 2; therefore, $\mathcal{S}$ must have delivered the message $(\mathcal{F}\textsf{-input}_j, sid, \cdot)$ from $\tilde{P}_j$ to $\mathcal{F}$ (in the external process) per Item 2. $\mathcal{S}$ verifies that the Round-1 message from $\mathcal{F}_{\text{ZK}}^{1,P_i \to P_j}$ to P_j (in the internal simulation) has been delivered per Item 2; if not, $\mathcal{S}$ does nothing. Otherwise, we are guaranteed that $\mathcal{S}$ has delivered the message $(\mathcal{F}\textsf{-input}_i, sid, \cdot)$ from $\tilde{P}_i$ to $\mathcal{F}$ (in the external process) per Item 2 as well. As both inputs were delivered to $\mathcal{F}$ and $\mathcal{F}$ has been subsequently activated (in the external process), $\mathcal{F}$ has written a message $(\mathcal{F}\textsf{-output}_j, sid, \perp)$ for $\tilde{P}_j$ on its outgoing communication tape (in the external process). $\mathcal{S}$ delivers $(\mathcal{F}\textsf{-output}_j, sid, \perp)$ from $\mathcal{F}$ to $\tilde{P}_j$ (in the external process).

After performing one of the above (if any), $\mathcal{S}$ copies the output value written by $\mathcal{A}$ on its output tape to $\mathcal{S}$'s own output tape, and ends its activation.

This completes the description of $\mathcal{S}$. All that is left to be shown is that no environment $\mathcal{Z}$ can tell with a non-negligible probability whether it is interacting with $\mathcal{A}$

and P_1, P_2 running $\pi_{\mathcal{F}}$ in the $\mathcal{F}_{\text{ZK}}$-hybrid model or with $\mathcal{S}$ and $\tilde{P}_1, \tilde{P}_2$ in the ideal process for $\mathcal{F}$. We focus on the case where $\mathcal{A}$ corrupts one of the parties, say P_2. The case where $\mathcal{A}$ corrupts P_1 is symmetric, and the cases where either or neither of the parties are corrupted are straightforward. Formally, we claim the following:

Lemma 4.2.2. *Let $\mathcal{A}$ be a (static) adversary that corrupts P_2 only, and let $\mathcal{S}$ be as above. Then for any $\mathcal{Z}$,*

$$EXEC^{\mathcal{F}_{ZK}}_{\pi_{\mathcal{F}},\mathcal{A},\mathcal{Z}} \overset{c}{\approx} IDEAL_{\mathcal{F},\mathcal{S},\mathcal{Z}}. \tag{4.1}$$

Loosely speaking, when P_2 only is corrupted, the following differences between a real-life execution of $\pi_{\mathcal{F}}$ among P_1, P_2 in the $\mathcal{F}_{\text{ZK}}$-hybrid model and the ideal process for $\mathcal{F}$ among $\tilde{P}_1, \tilde{P}_2$ may be noted: (1) in the former, P_1 computes its output based on a "garbled circuit" and obliviously-transferred input-wire labels corresponding to its input, received in the second round of the protocol, while in the latter, $\tilde{P}_1$ receives its output from $\mathcal{F}$ based on the value y that $\mathcal{S}$ obtained while simulating $\mathcal{F}_{\text{ZK}}^{1,P_2\to P_1}$ for the first round of the protocol; (2) in the former, the first round message from $F_{\text{ZK}}^{1,P_1\to P_2}$ to P_2 contains values $w_i = g^{c_i}$ where $c_i = a_i b_i$ when $x_i = 0$, $c_i = a_i b_i - 1$ when $x_i = 1$; while in the latter, the message (in the internal simulation) contains $w_i = g^{a_i b_i}$ for all i; (3) in the former, the second-round message from $\mathcal{F}_{\text{ZK}}^{2,P_1\to P_2}$ to P_2 contains values $K_{i,(1-y_i)}, C_{i,(1-y_i)}$ computed as in the specification of the protocol, while in the latter, those values (in the internal simulation) are chosen uniformly at random from $\mathcal{G}$; and (4) in the former, Yao_1 is used to compute the "garbled circuit" and input-wire labels for the second-round message from $\mathcal{F}_{\text{ZK}}^{2,P_1\to P_2}$ to P_2, while in the latter, $\mathsf{Yao\text{-}Sim}$ is used for that purpose, based on P_2's output from $\mathcal{F}(x, y)$, where y was obtained by $\mathcal{S}$ while simulating $F_{\text{ZK}}^{1,P_2\to P_1}$ for the first round of the protocol.

Nevertheless, we claim that Equation 4.1 holds, based on (1) the correctness of Yao's "garbled circuit" technique, the correctness of the oblivious transfer protocol and the enforcement of parties entering the two rounds of the protocol with a consistent input; (2) the hardness of the DDH assumption for $\mathsf{GroupGen}$; (3) the uniformity of $K_{i,(1-y_i)}, C_{i,(1-y_i)}$ per $\pi_{\mathcal{F}}$ in $\mathcal{G}$; and (4) the security Yao's construction. We proceed with a formal proof.

Proof (of lemma). We prove the lemma by defining an intermediate sequence of probabilistic games between the distributions in Equation 4.1 and showing that any two subsequent games in the sequence are (at most) computationally-indistinguishable.

GAME$_0$. Let GAME$_0$ denote $\text{EXEC}^{\mathcal{F}_{\text{ZK}}}_{\pi_{\mathcal{F}},\mathcal{A},\mathcal{Z}}$.

GAME$_1$. Let GAME$_1 \overset{\text{def}}{=} \text{EXEC}^{\left(\substack{\mathcal{F}_{\text{ZK}}^{1,P_1\to P_2},\ \mathcal{F}_{\text{ZK-pass}}^{1,P_2\to P_1}\\ \mathcal{F}_{\text{ZK}}^{2,P_1\to P_2},\ \mathcal{F}_{\text{ZK}}^{2,P_2\to P_1}}\right)}_{\pi_1,\mathcal{A},\mathcal{Z}}$, where:

- $\mathcal{F}_{\text{ZK-pass}}^{1,P_2\to P_1}$ acts precisely like $\mathcal{F}_{\text{ZK}}^{1,P_2\to P_1}$, except that for an incoming message

  $$(\mathsf{ZK\text{-}prover}, sid \circ 1 \circ P_2, \hat{x}, \hat{w})$$

  from P_2 such that $(\hat{x}, \hat{w}) \in R_1$, it sends $(\mathsf{ZK\text{-}proof}, sid \circ 1 \circ P_2, \hat{x})$ *along with* $\hat{w}$ to P_1 and the adversary.

- π_1 is identical to $\pi_{\mathcal{F}}$, except that here the parties use $\mathcal{F}_{\text{ZK-pass}}^{1,P_2\to P_1}$ (instead of $\mathcal{F}_{\text{ZK}}^{1,P_2\to P_1}$) for first-round communication originating from P_2. In addition: (1) upon receiving the first-round message

  $$(\mathsf{ZK\text{-}proof}, sid \circ 1 \circ P_2, (\{\bar{u}_i, \bar{v}_i, \bar{w}_i\}_i, (\mathcal{G}', q', g'), k'))$$

along with $(y, \{\bar{a}_i, \bar{b}_i\}_i)$ from $\mathcal{F}_{\text{ZK-pass}}^{1,P_2 \to P_1}$, P_1 performs the same checks as $\pi_{\mathcal{F}}$ instructs (on the onset of the second round), but also records y if they pass; and (2) upon receiving the second-round message from $\mathcal{F}_{\text{ZK}}^{2,P_2 \to P_1}$, P_1 performs the same checks as $\pi_{\mathcal{F}}$ instructs (in the output computation phase), but outputs $F_k^1(x, y)$ (where x is P_1's input).

We claim that GAME$_0$ and GAME$_1$ are identical. First, observe that the adversary has no access to the *contents* of a message for P_1 from either $\mathcal{F}_{\text{ZK}}^{1,P_2 \to P_1}$ in GAME$_0$ or from $\mathcal{F}_{\text{ZK-pass}}^{1,P_2 \to P_1}$ in GAME$_1$, and therefore cannot tell one (not containing a witness) from the other (containing the witness). The rest of the claim follows from the correctness of the OT protocol, the correctness of Yao's construction (cf. Section 4.1.4) and the enforcing of (the corrupted) P_2 to enter both rounds of the protocol with a consistent input. More precisely, note that the only difference left between the executions is in the computation of P_1's output. Observe that if the output computation phase is reached in GAME$_0$ (resp. in GAME$_1$), we are guaranteed that:

1. The values $\{u_i, v_i, w_i\}_i$ were constructed as specified in $\pi_{\mathcal{F}}$ (in both games);

2. $\mathcal{A}$ delivered a message

$$(\mathsf{ZK\text{-}prover}, sid \circ 1 \circ P_2, (\{\bar{u}_i, \bar{v}_i, \bar{w}_i\}_i, (\mathcal{G}, q, g), k), (y, \{\bar{a}_i, \bar{b}_i\}_i))$$

from (the corrupted) P_2 to $\mathcal{F}_{\text{ZK}}^{1,P_2 \to P_1}$ (resp. to $\mathcal{F}_{\text{ZK-pass}}^{1,P_2 \to P_1}$) such that

$$((\{\bar{u}_i, \bar{v}_i, \bar{w}_i\}_i, (\mathcal{G}, q, g), k), (y, \{\bar{a}_i, \bar{b}_i\}_i)) \in R_1; \tag{4.2}$$

3. $\mathcal{A}$ delivered

$$(\textsf{ZK-proof}, sid \circ 1 \circ P_2, (\{\bar{u}_i, \bar{v}_i, \bar{w}_i\}_i, (\mathcal{G}, q, g), k))$$

(along with $(y, \{\bar{a}_i, \bar{b}_i\}_i)$) in GAME$_1$) from $\mathcal{F}_{\text{ZK}}^{1,P_2 \to P_1}$ (resp. from $\mathcal{F}_{\text{ZK-pass}}^{1,P_2 \to P_1}$) to P_1;

4. $\mathcal{A}$ delivered a message

$$\left(\textsf{ZK-prover}, sid \circ 2 \circ P_2 \ , \ \begin{array}{c} \overline{\textsf{Circuit}}, \{\bar{K}_{i,\sigma}, \bar{C}_{i,\sigma}\}_{i,\sigma} \\ (\mathcal{G}, q, g), k, \{u_i, v_i, w_i\}_i \\ \{\bar{u}_i, \bar{v}_i, \bar{w}_i\}_i \end{array} \ , \ \begin{array}{c} y', \bar{\Omega}, \{\bar{Z}_{i,\sigma}\}_{i,\sigma} \\ \{\bar{r}_{i,\sigma}, \bar{s}_{i,\sigma}\}_{i,\sigma} \\ \{\bar{a}'_i, \bar{b}'_i\}_i \end{array} \right)$$

from (the corrupted) P_2 to $\mathcal{F}_{\text{ZK}}^{2,P_2 \to P_1}$ such that

$$\left(\begin{array}{c} \overline{\textsf{Circuit}}, \{\bar{K}_{i,\sigma}, \bar{C}_{i,\sigma}\}_{i,\sigma} \\ (\mathcal{G}, q, g), k, \{u_i, v_i, w_i\}_i \\ \{\bar{u}_i, \bar{v}_i, \bar{w}_i\}_i \end{array} \ , \ \begin{array}{c} y', \bar{\Omega}, \{\bar{Z}_{i,\sigma}\}_{i,\sigma} \\ \{\bar{r}_{i,\sigma}, \bar{s}_{i,\sigma}\}_{i,\sigma} \\ \{\bar{a}'_i, \bar{b}'_i\}_i \end{array} \right) \in R_2; \quad \text{and} \tag{4.3}$$

5. $\mathcal{A}$ delivered

$$(\textsf{ZK-proof}, sid \circ 2 \circ P_2, (\overline{\textsf{Circuit}}, \{\bar{K}_{i,\sigma}, \bar{C}_{i,\sigma}\}_{i,\sigma}, (\mathcal{G}, q, g), k, \{u_i, v_i, w_i\}_i, \{\bar{u}_i, \bar{v}_i, \bar{w}_i\}_i))$$

from $\mathcal{F}_{\text{ZK}}^{2,P_2 \to P_1}$ to P_1.

(The above implicitly takes into account the consistency verifications that P_1 performs in both $\pi_{\mathcal{F}}$ and π_1 at the onset of the second round and prior to the output computation.)

We first note that in both executions, since for all i,

$$\bar{u}_i = g^{\bar{a}_i}, \quad \bar{v}_i = g^{\bar{b}_i}, \quad \bar{w}_i = \begin{cases} g^{\bar{a}_i \bar{b}_i} & y_i = 0 \\ g^{\bar{a}_i \bar{b}_i - 1} & \text{otherwise} \end{cases}$$

per Equation 4.2, and since for all i,

$$\bar{u}_i = g^{\bar{a}'_i}, \quad \bar{v}_i = g^{\bar{b}'_i}, \quad \bar{w}_i = \begin{cases} g^{\bar{a}'_i \bar{b}'_i} & y'_i = 0 \\ g^{\bar{a}'_i \bar{b}'_i - 1} & \text{otherwise} \end{cases}$$

per Equation 4.3 — it must be the case that $y' = y$.

Next, note that by Equation 4.3 we have that in both executions, $(\overline{\mathsf{Circuit}}, \{\bar{Z}_{i,\sigma}\}_{i,\sigma}) = \mathsf{Yao}_1(k, F^1_k, y'; \bar{\Omega})$. Furthermore, Equation 4.3 and the proper construction of the values $\{u_i, v_i, w_i\}_i$ guarantees that for any i, if $x_i = 0$,

$$\begin{aligned} \bar{Z}_i &\overset{\text{def}}{=} \bar{K}^{-b_i}_{i,x_i} \cdot \bar{C}_{i,x_i} \\ &= (u_i^{\bar{r}_{i,0}} \cdot g^{\bar{s}_{i,0}})^{-b_i} \cdot (w_i^{\bar{r}_{i,0}} \cdot v_i^{\bar{s}_{i,0}} \cdot \bar{Z}_{i,0}) \\ &= (g^{a_i \bar{r}_{i,0}} \cdot g^{\bar{s}_{i,0}})^{-b_i} \cdot (g^{a_i b_i \bar{r}_{i,0}} \cdot g^{b_i \bar{s}_{i,0}} \cdot \bar{Z}_{i,0}) = \bar{Z}_{i,0}, \end{aligned}$$

and if $x_i = 1$,

$$\begin{aligned} \bar{Z}_i &\overset{\text{def}}{=} \bar{K}^{-b_i}_{i,x_i} \cdot \bar{C}_{i,x_i} \\ &= (u_i^{\bar{r}_{i,1}} \cdot g^{\bar{s}_{i,1}})^{-b_i} \cdot ((g \cdot w_i)^{\bar{r}_{i,1}} \cdot v_i^{\bar{s}_{i,1}} \cdot \bar{Z}_{i,1}) \\ &= (g^{a_i \bar{r}_{i,1}} \cdot g^{\bar{s}_{i,1}})^{-b_i} \cdot (g^{a_i b_i \bar{r}_{i,1}} \cdot g^{b_i \bar{s}_{i,1}} \cdot \bar{Z}_{i,1}) = \bar{Z}_{i,1}; \end{aligned}$$

in other words, for any i, $\bar{Z}_i = \bar{Z}_{i,x_i}$. In GAME$_0$, P_1 outputs $\mathsf{Yao}_2(k, \overline{\mathsf{Circuit}}, \{\bar{Z}_i\}_i)$, which therefore equals $\mathsf{Yao}_2(k, \overline{\mathsf{Circuit}}, \{\bar{Z}_{i,x_i}\}_i)$. In GAME$_1$, P_1 outputs $F^1_k(x, y) = F^1_k(x, y')$. But by the correctness property of Yao's construction, these are equal.

GAME$_2$. Next, let GAME$_2 \overset{\text{def}}{=} \text{EXEC}_{\pi_2,\mathcal{A},\mathcal{Z}}^{\left(\substack{\mathcal{F}^{P_1\to P_2}_{\text{ZK-consult}},\ \mathcal{F}^{1,P_2\to P_1}_{\text{ZK-pass}} \\ \mathcal{F}^{2,P_1\to P_2}_{\text{ZK}},\ \mathcal{F}^{2,P_2\to P_1}_{\text{ZK}}}\right)}$, where:

- $\mathcal{F}_{\text{ZK-consult}}$ mimics the input/output behavior of $\mathcal{F}_{\text{ZK}}$, but is controlled by the prover and produces an on-the-fly "proof" for a statement $\hat{x}$ without ever seeing a witness $\hat{w}$ such that $(\hat{x}, \hat{w}) \in R$. In particular, on a (dummy) incoming message $(\textsf{ZK-prover}, sid, \cdot, \cdot)$, $\mathcal{F}_{\text{ZK-consult}}^{P_1 \to P_2}$ asks the prover P_1 for $\hat{x}$ and sends $(\textsf{ZK-proof}, sid, \hat{x})$ to the verifier P_2 and the adversary.

- π_2 is identical to π_1, except that here the parties use $\mathcal{F}_{\text{ZK-consult}}^{P_1 \to P_2}$ for first-round communication originating at P_1, as follows. In the first round, P_1 sends a (dummy) message $(\textsf{ZK-prover}, sid \circ 1 \circ P_1, \bot, \bot)$ to $\mathcal{F}_{\text{ZK-consult}}^{P_1 \to P_2}$. When the latter asks for a statement to be "proven", P_1 provides:

$$\hat{x} = (\{u_i, v_i, w_i\}_i, (\mathcal{G}, q, g), k)),$$

computed precisely as in π_1. In effect, P_1 in π_2 provides the functionality with the exact same statement to be proven as in π_1, but without the witness.

We claim that GAME$_1$ is identical to GAME$_2$. To see this, note that (1) the adversary does not have access to the *contents* of the first-round message from P_1 to either $\mathcal{F}_{\text{ZK}}^{1,P_1 \to P_2}$ in GAME$_1$ or to $\mathcal{F}_{\text{ZK-consult}}^{P_1 \to P_2}$ in GAME$_2$, and therefore cannot distinguish one (with contents) from the other (with dummy contents); and (2) the adversary cannot distinguish the outgoing message $\mathcal{F}_{\text{ZK}}^{1,P_1 \to P_2}$ produces in GAME$_1$ from the message $\mathcal{F}_{\text{ZK-consult}}^{P_1 \to P_2}$ produces in GAME$_2$, as they are identically distributed (in GAME$_1$, the message contains the statement $\hat{x}$, where $\hat{x}$ was provided with $\hat{w}$ such that $(\hat{x}, \hat{w}) \in R_1$; in GAME$_2$, the message contains a statement computed in the exact same way, and is therefore identically distributed). We conclude that Z cannot distinguish between the games, as claimed.

GAME$_3$. Let GAME$_3 \overset{\text{def}}{=} \text{EXEC}_{\pi_3, \mathcal{A}, \mathcal{Z}}^{\left(\substack{\mathcal{F}_{\text{ZK-consult}}^{P_1 \to P_2},\ \mathcal{F}_{\text{ZK-pass}}^{1,P_2 \to P_1} \\ \mathcal{F}_{\text{ZK}}^{2,P_1 \to P_2},\ \mathcal{F}_{\text{ZK}}^{2,P_2 \to P_1}}\right)}$, where

- π_3 is identical to π_2, except that in the first round, when $\mathcal{F}_{\text{ZK-consult}}^{1,P_1 \to P_2}$ asks P_1

for a first-round statement to be "proven", P_1 provides

$$\hat{x} = (\{u_i, v_i, w_i\}_i, (\mathcal{G}, q, g), k),$$

where the values $\{u_i, v_i, w_i\}_i$ are computed as in π_2, except that P_1 sets $w_i = g^{a_i b_i}$ *for all* i (as opposed to setting w_i to $g^{a_i b_i}$ when $x_i = 0$ and to $g^{a_i b_i - 1}$ when $x_i = 1$, as is done in π_2).

We now claim that $\text{GAME}_2 \overset{c}{\approx} \text{GAME}_3$. By a standard hybrid argument, it suffices to show that $\text{GAME}_2^{(i^*-1)} \overset{c}{\approx} \text{GAME}_2^{i^*}$ for some $0 < i^* \leq k$, where $\text{GAME}_2^{(i^*-1)} \overset{\text{def}}{=} \text{EXEC}_{\pi_2^{(i^*-1)}, \mathcal{A}, \mathcal{Z}}^{\left(\begin{smallmatrix}\mathcal{F}_{\text{ZK-consult}}^{P_1 \to P_2}, & \mathcal{F}_{\text{ZK-pass}}^{1, P_2 \to P_1} \\ \mathcal{F}_{\text{ZK}}^{2, P_1 \to P_2}, & \mathcal{F}_{\text{ZK}}^{2, P_2 \to P_1}\end{smallmatrix}\right)}$, $\text{GAME}_2^{i^*} \overset{\text{def}}{=} \text{EXEC}_{\pi_2^{i^*}, \mathcal{A}, \mathcal{Z}}^{\left(\begin{smallmatrix}\mathcal{F}_{\text{ZK-consult}}^{P_1 \to P_2}, & \mathcal{F}_{\text{ZK-pass}}^{1, P_2 \to P_1} \\ \mathcal{F}_{\text{ZK}}^{2, P_1 \to P_2}, & \mathcal{F}_{\text{ZK}}^{2, P_2 \to P_1}\end{smallmatrix}\right)}$, and:

- π_2^j is identical to π_2, aside from the following. Let $i_1, \ldots i_j$ be the first j coordinates on which x, the input to P_1, is equal to 1. When $\mathcal{F}_{\text{ZK-consult}}^{P_1 \to P_2}$ asks P_1 for a first-round statement to be "proven", P_1 provides

  $$\hat{x} = (\{u_i, v_i, w_i\}_i, (\mathcal{G}, q, g), k),$$

  where the values $\{u_i, v_i, w_i\}_i$ are computed as in π_2, except that for any $\ell \in \{i_1, \ldots i_j\}$, P_1 sets $w_\ell = g^{a_\ell b_\ell}$ (as opposed to setting $w_\ell = g^{a_\ell b_\ell - 1}$ as in π_2).

Intuitively, $\text{GAME}_2^{(i^*-1)} \overset{c}{\approx} \text{GAME}_2^{i^*}$ by the hardness of the DDH problem for $\mathsf{GroupGen}$. Specifically, assume $\mathcal{Z}, \mathcal{A}$ attempt to distinguish the said games; we construct an algorithm D that attacks the hardness of the DDH problem for $\mathsf{GroupGen}$. Algorithm D on input $(k, z, \mathcal{G}, q, g, u, v, w)$ sets an internal (simulated) CRS to $(\mathcal{G}, q, g)$. It then invokes $\mathcal{Z}(z)$ and $\mathcal{A}$, simulating for them the functionalities $\mathcal{F}_{\text{ZK-consult}}^{P_1 \to P_2}, \mathcal{F}_{\text{ZK-pass}}^{1, P_2 \to P_1}$,

$\mathcal{F}_{\text{ZK}}^{2,P_1\to P_2}$ and $\mathcal{F}_{\text{ZK}}^{2,P_2\to P_1}$ and an honest P_1 running $\pi_2^{(i^*-1)}$ in a straightforward manner, with the following caveat:

- In the simulation of $\mathcal{F}_{\text{ZK-consult}}^{P_1\to P_2}$, when $\mathcal{F}_{\text{ZK-consult}}^{P_1\to P_2}$ asks (the simulated) P_1 for a first-round statement to be "proven", D provides

$$\hat{x} = (\{u_i, v_i, w_i\}_i, (\mathcal{G}, q, g), k),$$

where the values $\{u_i, v_i, w_i\}_i$ are computed precisely as in $\pi_2^{(i^*-1)}$ for all i, except for the i^*'s coordinate on which x is 1; call this coordinate i_{i^*}. For this coordinate, D sets $u_{i_{i^*}} = u, v_{i_{i^*}} = v, w_{i_{i^*}} = w$ (recall that u, v, w are inputs to D).

D outputs whatever $\mathcal{Z}$ outputs at the end of the simulation. Note that

$$\left\{ \begin{array}{c} (\mathcal{G}, q, g) \xleftarrow{R} \mathsf{GroupGen}(k); a, b \xleftarrow{R} \mathbb{Z}_q : \\ D(k, z, \mathcal{G}, q, g, g^a, g^b, g^{ab-1}) \end{array} \right\}_{k\in\mathbb{N}, z\in\{0,1\}^*} = \text{EXEC}_{\pi_2^{(i^*-1)}, \mathcal{A}, \mathcal{Z}}^{\left(\begin{array}{c}\mathcal{F}_{\text{ZK-consult}}^{P_1\to P_2},\ \mathcal{F}_{\text{ZK-pass}}^{1,P_2\to P_1} \\ \mathcal{F}_{\text{ZK}}^{2,P_1\to P_2},\ \mathcal{F}_{\text{ZK}}^{2,P_2\to P_1}\end{array}\right)},$$

and that

$$\left\{ \begin{array}{c} (\mathcal{G}, q, g) \xleftarrow{R} \mathsf{GroupGen}(k); a, b \xleftarrow{R} \mathbb{Z}_q : \\ D(k, z, \mathcal{G}, q, g, g^a, g^b, g^{ab}) \end{array} \right\}_{k\in\mathbb{N}, z\in\{0,1\}^*} = \text{EXEC}_{\pi_2^{i^*}, \mathcal{A}, \mathcal{Z}}^{\left(\begin{array}{c}\mathcal{F}_{\text{ZK-consult}}^{P_1\to P_2},\ \mathcal{F}_{\text{ZK-pass}}^{1,P_2\to P_1} \\ \mathcal{F}_{\text{ZK}}^{2,P_1\to P_2},\ \mathcal{F}_{\text{ZK}}^{2,P_2\to P_1}\end{array}\right)}.$$

By Claim 4.1.4, it follows that $\text{GAME}_2^{(i^*-1)} \overset{c}{\approx} \text{GAME}_2^{i^*}$.

GAME$_4$. Let $\text{GAME}_4 \overset{\text{def}}{=} \text{EXEC}_{\pi_4, \mathcal{A}, \mathcal{Z}}^{\left(\begin{array}{c}\mathcal{F}_{\text{ZK-consult}}^{P_1\to P_2}[1],\ \mathcal{F}_{\text{ZK-pass}}^{1,P_2\to P_1} \\ \mathcal{F}_{\text{ZK-consult}}^{P_1\to P_2}[2],\ \mathcal{F}_{\text{ZK}}^{2,P_2\to P_1}\end{array}\right)}$, where:

- $\mathcal{F}_{\text{ZK-consult}}^{P_1\to P_2}[i]$ is an instance of $\mathcal{F}_{\text{ZK-consult}}^{P_1\to P_2}$, as defined in GAME$_2$; here we use two distinct instances of the same functionality.

- π_4 is identical to π_3 (in particular, uses instance $\mathcal{F}_{\text{ZK-consult}}^{P_1\to P_2}[1]$ for first-round communication originating at P_1, as before), except that the parties here use

$\mathcal{F}^{P_1 \to P_2}_{\text{ZK-consult}}[2]$ for second-round communication originating at P_1, as follows. In the second round, P_1 sends a (dummy) message $(\mathsf{ZK\text{-}prover}, sid \circ 2 \circ P_1, \perp, \perp)$ to $\mathcal{F}^{P_1 \to P_2}_{\text{ZK-consult}}[2]$. When the latter asks for a statement to be "proven", P_1 provides:

$$\hat{x} = (\mathsf{Circuit}, \{K_{i,\sigma}, C_{i,\sigma}\}_{i,\sigma}, (\mathcal{G}, q, g), k, \{\bar{u}_i, \bar{v}_i, \bar{w}_i\}_i, \{u_i, v_i, w_i\}),$$

where all the above are computed exactly as in π_3. In effect, P_1 in π_4 provides the functionality with the exact same second-round statement to be proven as in π_3, but without the witness.

We claim that GAME$_3$ is identical to GAME$_4$. To see this, note that (1) the adversary does not have access to the *contents* of the second-round message from P_1 to either $\mathcal{F}^{2,P_1 \to P_2}_{\text{ZK}}$ in GAME$_3$ or to $\mathcal{F}^{P_1 \to P_2}_{\text{ZK-consult}}[2]$ in GAME$_4$, and therefore cannot distinguish one (with contents) from the other (with dummy contents); and (2) the adversary cannot distinguish the outgoing message $\mathcal{F}^{2,P_1 \to P_2}_{\text{ZK}}$ produces in GAME$_3$ from the message $\mathcal{F}^{P_1 \to P_2}_{\text{ZK-consult}}[2]$ produces in GAME$_4$, as they are identically distributed (in GAME$_3$, the message contains the statement $\hat{x}$, where $\hat{x}$ was provided with $\hat{w}$ such that $(\hat{x}, \hat{w}) \in R_2$; in GAME$_4$, the message contains a statement computed in the exact same way, and is therefore identically distributed). We conclude that Z cannot distinguish between the games, as claimed.

GAME$_5$. Define GAME$_5 \overset{\text{def}}{=} \text{EXEC}^{\left(\substack{\mathcal{F}^{P_1 \to P_2}_{\text{ZK-consult}}[1],\ \mathcal{F}^{1,P_2 \to P_1}_{\text{ZK-pass}} \\ \mathcal{F}^{P_1 \to P_2}_{\text{ZK-consult}}[2],\ \mathcal{F}^{2,P_2 \to P_1}_{\text{ZK}}}\right)}_{\pi_5, \mathcal{A}, \mathcal{Z}}$, where:

- π_5 is identical to π_4, except that after receiving (and testing) first-round message

$$(\mathsf{ZK\text{-}proof}, sid \circ 1 \circ P_2, (\{\bar{u}_i, \bar{v}_i, \bar{w}_i\}_i, (\mathcal{G}, q, g), k)),$$

along with $(y, \{\bar{a}_i, \bar{b}_i\}_i)$ from $\mathcal{F}^{1,P_2 \to P_1}_{\text{ZK-pass}}$, and upon being asked by $\mathcal{F}^{P_1 \to P_2}_{\text{ZK-consult}}[2]$ for a second-round statement, P_1 provides $\mathcal{F}^{P_1 \to P_2}_{\text{ZK-consult}}[2]$ with

$$\hat{x}_2 = (\mathsf{Circuit}, \{K_{i,\sigma}, C_{i,\sigma}\}_{i,\sigma}, (\mathcal{G}, q, g), k, \{\bar{u}_i, \bar{v}_i, \bar{w}_i\}_i, \{u_i, v_i, w_i\}_i),$$

where all the components of $\hat{x}_2$ are computed precisely as in π_4, except that P_1, for every i, sets $K_{i,1-y_i}, C_{i,1-y_i}$ to be elements chosen uniformly at random from $\mathcal{G}$.

We claim that GAME$_4$ is identical to GAME$_5$. We first note that in both executions, if the contents of P_2's first-round message verify on R_1, it is the case that

$$K_{i,1-y_i} = g^{\bar{a}_i \cdot r_{(i,1-y_i)} + s_{(i,1-y_i)}}$$

$$C_{i,1-y_i} = g^{\bar{c}_{(i,1-y_i)} \cdot r_{(i,1-y_i)} + \bar{b}_i \cdot s_{(i,1-y_i)}} \cdot Z_{i,1-y_i}$$

where

$$\bar{c}_{(i,1-y_i)} = \begin{cases} \bar{a}_i\bar{b}_i + 1 & y_i = 0 \\ \bar{a}_i\bar{b}_i - 1 & \text{otherwise} \end{cases} \neq \bar{a}_i\bar{b}_i.$$

We claim that for for any $\bar{a}, \bar{b}, \bar{c}, Z \in \mathcal{G}$ such that $\bar{c} \neq \bar{a}\bar{b}$, the group elements $K \overset{\text{def}}{=} g^{\bar{a}r+s}$ and $C \overset{\text{def}}{=} g^{\bar{c}r+\bar{b}s} \cdot Z$ are independent and uniformly distributed in $\mathcal{G}$ when r, s are picked uniformly at random from $\mathcal{G}$. To see this, fix some $\alpha, \beta \in \mathcal{G}$, and consider the following system of equations in r, s:

$$\begin{bmatrix} g^{\bar{a}r+s} & = \alpha \\ g^{\bar{c}r+\bar{b}s} \cdot Z & = \beta \end{bmatrix}.$$

Rearranging, we have:

$$\begin{bmatrix} \bar{a}r + s & = \log_g \alpha \\ \bar{c}r + \bar{b}s & = \log_g \beta - \log_g Z \end{bmatrix},$$

which we write equivalently as

$$\mathbf{B} \cdot \begin{pmatrix} r \\ s \end{pmatrix} = \begin{pmatrix} \log_g \alpha \\ \log_g \beta - \log_g Z \end{pmatrix}$$

where $\mathbf{B} = \begin{pmatrix} \bar{a} & 1 \\ \bar{c} & \bar{b} \end{pmatrix}$. Note that $\det(\mathbf{B}) = \bar{a}\bar{b} - \bar{c} \neq 0$ when $\bar{c} \neq \bar{a}\bar{b}$. Therefore, the system has a unique solution in r, s for each choice of α, β. It follows that when r, s are chosen uniformly at random from $\mathcal{G}$, $\alpha = K$ and $\beta = C$ are uniformly distributed in $\mathcal{G}$.

Therefore, the adversary cannot distinguish the output message produced by $\mathcal{F}^{P_1 \to P_2}_{\text{ZK-consult}}[2]$ in GAME_4, where the $\{K_{i,1-y_i}, C_{i,1-y_i}\}_i$ are computed as in π_4, from the output message produced by $\mathcal{F}^{P_1 \to P_2}_{\text{ZK-consult}}[2]$ in GAME_5, where the $\{K_{i,1-y_i}, C_{i,1-y_i}\}_i$ are chosen uniformly at random from $\mathcal{G}$. It follows that $\text{GAME}_4, \text{GAME}_5$ are identical.

GAME_6. Let $\text{GAME}_6 \overset{\text{def}}{=} \text{EXEC}^{\left(\substack{\mathcal{F}^{P_1 \to P_2}_{\text{ZK-consult}}[1],\ \mathcal{F}^{1, P_2 \to P_1}_{\text{ZK-pass}} \\ \mathcal{F}^{P_1 \to P_2}_{\text{ZK-consult}}[2],\ \mathcal{F}^{2, P_2 \to P_1}_{\text{ZK}}}\right)}_{\pi_6, \mathcal{A}, \mathcal{Z}}$, where:

- π_6 is identical to π_5, except that when P_1 is asked by $\mathcal{F}^{P_1 \to P_2}_{\text{ZK-consult}}[2]$ for a second-round statement, P_1 computes $v \leftarrow \mathcal{F}^2_k(x, y)$, computes $(\mathsf{Circuit}, \{Z_i\}_i) \overset{R}{\leftarrow} \mathsf{Yao\text{-}Sim}(k, \mathcal{F}^2_k, y, v)$, and sets $Z_{i,y_i} = Z_i$ for every i; it continues preparing the statement as in π_5.

It is straightforward to show that $\mathrm{GAME}_5 \overset{c}{\approx} \mathrm{GAME}_6$ based on the security of Yao's garbled circuit (cf. Section 4.1.4).

GAME$_7$. Finally, note that GAME_6 is identical to $\mathrm{GAME}_7 \overset{\text{def}}{=} \mathrm{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$. This is so, because the differences between the executions are conceptual: in the former, P_1 computes $\mathcal{F}^2_k(x,y)$ by itself, while in the latter it receives it from $\mathcal{F}$; in the former, P_1 computes $F^1_k(x,y)$ itself, while in the latter $\tilde{P}_1$ receives it from $\mathcal{F}$; in addition, in the former, the parties use $\mathcal{F}^{P_1 \to P_2}_{\text{ZK-consult}}[1], \mathcal{F}^{1,P_2 \to P_1}_{\text{ZK-pass}}, \mathcal{F}^{P_1 \to P_2}_{\text{ZK-consult}}[2]$ and $\mathcal{F}^{2,P_2 \to P_1}_{\text{ZK}}$, while in the latter $\mathcal{S}$ perfectly simulates these functionalities. We conclude that $\mathrm{EXEC}^{\mathcal{F}_{\text{ZK}}}_{\pi_{\mathcal{F}},\mathcal{A},\mathcal{Z}} \overset{c}{\approx} \mathrm{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$ as required. ■

4.2.3 Round Optimality

It is almost immediate that two rounds are necessary for two-party computation, even if both parties are allowed to speak simultaneously, under any reasonable definition of security. Loosely speaking, consider a candidate single-round protocol for a functionality that provides output to one of the parties, say P_2. Since (an honest) P_1 sends its message independently of P_2's input, P_2 can (honestly) run its output-computation segment of the protocol on the incoming message multiple times using inputs of its choice, and learn the output of the functionality on each. This clearly violates security except for functionalities that do not depend on P_2's input.

More formally and in the context of UC security, consider the functionality $\mathcal{F}_=$, which on input a pair of two-bit strings $x, y \in \{0,1\}^2$, provides P_2 with output

1 if $x = y$, 0 otherwise. Assume π UC-realizes $\mathcal{F}_=$ in a single round. Let π^{P_1} be the procedure in π that takes P_1's input x and a security parameter k and outputs P_1's outgoing message m; let π^{P_2} be the procedure in π that takes P_2's input y, an incoming message m and a security parameter k, and computes P_2's output value v. As π UC-realizes $\mathcal{F}_=$, it must be the case that for any x, y and with all but negligible probability in k, if $m \overset{R}{\leftarrow} \pi^{P_1}(x, k)$ and $v \overset{R}{\leftarrow} \pi^{P_2}(y, m, k)$, then $v = \mathcal{F}_=(x, y)$ (by considering a benign adversary that does not corrupt any party and delivers all messages as prescribed by π).

Consider an environment $\mathcal{Z}$ which picks x uniformly at random from $\{0,1\}^2$ and provides x as input to P_1. Consider an adversary $\mathcal{A}$, participating in a real-life execution of π, that acts as follows. $\mathcal{A}$ corrupts P_2 on the onset of the execution. On an incoming message m from P_1, $\mathcal{A}$ computes $\pi^{P_2}(y, m, k)$ on *all* four strings $y \in \{0,1\}^2$, and outputs (the lexicographically first) y on which the computation produces 1. Note that by the above, with all but negligible probability, $\mathcal{A}$ outputs x. We claim that for any ideal-process adversary $\mathcal{S}$, $\mathcal{Z}$ may distinguish a real-life execution of π in the presence of $\mathcal{A}$ from the ideal process involving $\mathcal{S}$ and $\mathcal{F}_=$. To see this, observe that $\mathcal{S}$'s probability of outputting x is at most $1/4$, as its view in the ideal process is independent of x.

For the setting in which parties need to take turns in sending their protocol messages, note that a two-round protocol that provides output to both participating parties implies a one-round protocol that provides output to one. Proceed with the argument above to obtain that three rounds are necessary for this setting.

4.3 Application to Universally-Composable Blind Signatures

In this section, we briefly discuss how the simpler of our protocols can be used to construct a round-optimal (i.e., two-round) UC-secure blind signature scheme in the CRS model. We begin with a quick recap of the definitions. Roughly speaking, a blind signature scheme should guarantee *unforgeability* and *blindness*. The first property requires that if a malicious user interacts with the honest signer for a total of ℓ executions of the protocol (in an arbitrarily-interleaved fashion), then the user should be unable to output valid signatures on $\ell + 1$ distinct messages. (A stronger requirement called *strong unforgeability* requires that the user cannot even output $\ell + 1$ distinct signatures on $\ell + 1$ possibly-repeating messages.) Blindness requires, very informally, that a malicious signer cannot "link" a particular execution of the protocol to a particular user *even after observing the signature obtained by the user.* This is formalized (see, e.g., [Fis06]) by a game in which the signer interacts with two users in an order determined by a randomly-chosen selector bit b, and should be unable to guess the value of b (with probability significantly better than 1/2) even after being given the signatures computed by these two users. This definition also allows the malicious signer to generate its public key in any manner (and not necessarily following the legitimate key-generation algorithm).

The above represent the "classical" definitions of security for blind signatures. Fischlin [Fis06] formally defines a *blind signature functionality* in the UC framework. He also gives a two-round protocol realizing this functionality. Interestingly, one of the motivations cited in [Fis06] for *not* relying on the generic results of [CLOS02] is the desire to obtain a round-optimal protocol.

Assume we have a (standard) signature scheme $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$, and consider the (randomized) functionality $\mathcal{F}_{\mathsf{sign}}(SK, m) = \mathsf{Sign}_{SK}(m)$. Contrary to what might be a naive first impression, secure computation of this functionality does *not* (in general) yield a secure blind signature scheme! (See also [JLO97].) Specifically, the problem is that the signer may use *different* secret keys SK, SK' in different executions of the protocol. Furthermore, the public key may be set up in such a way that each secret key yields a valid signature. Then, upon observing the signatures computed by the users, the signer may be able to tell which key was used to generate each signature, thereby violating the users' anonymity.

Juels, Luby, and Ostrovsky [JLO97] suggest a relatively complex method for handling this issue. We observe that a much simpler solution is possible by simply forcing the signer to use *a fixed signing key in every execution of the protocol.* This is done in the following way: To generate a public/secret key, the signer first computes $(PK, SK) \leftarrow \mathsf{Gen}(1^k)$. It then computes a (perfectly-binding) commitment $\mathsf{com} = \mathsf{Com}(SK; \omega)$ to SK using randomness ω. The public key is PK, com and the secret key contains SK and ω.

Define functionality $\mathcal{F}^*_{\mathsf{sign}}((SK, \omega), (\mathsf{com}, m))$ as follows: if $\mathsf{Com}(SK; \omega) = \mathsf{com}$, then the second party receives output $\mathsf{Sign}_{SK}(m)$ (when Sign is randomized, the functionality chooses a uniform random tape for computing this signature). Otherwise, the second party receives output $\perp$. The first party receives no output in either case.

It is not hard to see that a protocol for secure computation of $\mathcal{F}^*_{\mathsf{sign}}$ yields a secure blind signature scheme; using a UC two-party computation protocol for $\mathcal{F}^*_{\mathsf{sign}}$ gives a UC blind signature scheme. Using the simple two-round protocol constructed in Section 4.2.1, and noticing that only one party receives output here, we thus obtain a two-round UC blind signature scheme.

4.4 Application to Evaluation of Trust Policies on Sets of Credentials

Trust establishment between a server and a client typically involves an interaction that enables the server to evaluate his policy on a subset of the client's credentials. It is often desirable that the interaction provide *privacy* guarantees to the participating parties. In rough terms, a procedure that limits the amount of information disclosed to the client about the server's policy is said to provide server privacy guarantees; similarly, a procedure that limits the amount of information disclosed to the server about the client's credentials is said to provide client privacy guarantees.

Note that the evaluation of a server's policy on a client's set of credentials can be cast as an instance of two-party computation (by having the policy as the server's input, the credentials as the client's, and the functionality a circuit that evaluates one on the other). Applying our main construction yields a solution that provides full privacy guarantees to both the client and the server in a minimal number of rounds while preserving security under general composition. On the down side, the approach may entail relatively high communication complexity, as a circuit encoding the verification of cryptographic credentials — to be garbled and sent per our construction — may be of large size. In this section, we briefly compare this approach to other solutions suggested in the literature.

Trust-Negotiation. In the trust-negotiation setting, the server and the client both maintain policies that regulate access to their resources (be it credentials or the policies themselves). Trust-negotiation techniques (see [BS00, SWY01, YWS01, YW03, WL04] and references therein) involve strategies for the gradual disclosure of the resources, such that no resource is revealed unless its access-control policy has been satisfied. Resources that are cleared are fully disclosed, and so these techniques do not provide full privacy guarantees to either party.

Hidden Credentials. *Hidden Credential* schemes [HBS03, BHS04] allow a client to decrypt an encrypted server resource when the client holds credentials that satisfy the server's access-policy; the credentials are never revealed to the server (in fact, the server never learns whether access to the resource has been granted or not). The schemes thus provide full client-privacy. However, the client may learn information about the structure of the access policy (expressed as a boolean formula with threshold gates, in the most general case), even if its credentials do not satisfy it. The schemes therefore provide only partial server-privacy. Specifying policies that ask for a credential attribute to fall within a *range* of values is inefficient in current Hidden Credential schemes. The schemes are based on Identity-Based Encryption (IBE) (see [BF03] and references therein), consider the problem in a stand-alone setting, are proven secure in Random Oracle (RO) model and are communication- and round efficient.

Oblivious Envelopes. *Oblivious Envelopes* allow a client to decrypt an encrypted server resource when the client holds a credential that attests to its compliance with an agreed upon, *public* policy (as such, the schemes offer no server privacy guarantees); as in Hidden Credential schemes, the server is oblivious to the client's access attempts — the schemes provide full client privacy. Signature-based constructions [LDB03, NT05] allow the use of general policies (expressed as boolean circuits). A construction based on Pederson commitments [LL05a] only allows for simple policies that may compare an attribute in a credential to specified values. An improved construction based on integer commitments [LL06] allows the (efficient) use of any predicate for which an (efficient) zero-knowledge proof system exists. All the above constructions consider the problem in the stand-alone setting, are proven secure in the RO model and are communication- and round efficient.

Policy-Based Encryption. *Policy-Based Encryption* schemes [BM05, BMC06] allow a server, holding a resource and a corresponding *public* access-control policy, to encrypt the resource such that only a client holding credentials that satisfy the policy may decrypt it. As in Oblivious Envelopes, the schemes provide full client privacy and no server privacy. The constructions allow the use of general policies (expressed as monotone boolean formulae), are based on bilinear-pairings over elliptic curves, are considered in the stand-alone setting, proven secure in the RO model and are communication- and round efficient.

Solutions Based on Generic Secure Two-Party Computation Protocols. Casting the problem as an instance of secure two-party computation, one may use generic protocols to obtain solutions that provide full server and client privacy, but entail a relatively high communication complexity. Specifically, in the stand-alone setting, applying the protocol of Katz and Yung [KO04] gives a five-round solution based on general assumptions; while applying the protocol of Cachin et. al. [CCKM00] gives a three-round solution based on the DDH assumption and the availability of a CRS in the network. In the UC setting and under the CRS assumption, applying the protocol of Canetti et. al. [CLOS02] gives a non-constant round solution secure against adaptive adversaries; applying the protocol of Jarecki and Shmatikov [JS07] yields a five-round solution (three-round, if one further assumes the RO model) based on the strong RSA and the Decisional Composite Residuosity (DCR) assumptions, secure against static corruption. (All the constant-rounds solutions above may save a round of communication if only the server is to receive output.)

Improving the Efficiency of Solutions Based on Generic Secure Two-Party Computation Protocols. Several solutions aim at reducing the high communication complexity incurred when applying generic secure two-party computation protocols to our problem by removing credential verification from the two-party functionality to be computed and dealing with it separately. All consider the problem in the stand-alone setting. *Certified Input Private Policy Evaluation* [LL05b] relies on commitment-based certificates; it provides full server privacy against malicious adversaries, but full client privacy only for weak-honest adversaries (for malicious adversaries, some client information is leaked; leakage is detectable). A similar solution based on IBE certificates is given in [FAL06]. Both solutions are round efficient and offer improved communication complexity as compared with solutions based on generic secure two-party computation protocols. The protocol proposed in [FLA06] aims at providing a full trust negotiation mechanism with full privacy guarantees to both participating parties. The solution is based on IBE certificates and homeomorphic encryption and offers improved communication efficiency as compared with solutions based on generic secure two-party computation. However, round complexity is proportional to the sizes of the credential sets held by the parties.

Bibliography

[AG99] M. Abadi and A.D. Gordon. A Calculus for Cryptographic Protocols: The Spi Calculus. In *Information and Computation*, vol. 148(1), pp. 1–70, 1999.

[AIR01] W. Aiello, Y. Ishai and O. Reingold. Priced Oblivious Transfer: How to Sell Digital Goods. In *Advances in Cryptology — EUROCRYPT 2001*, LNCS 2045, pp. 119–135, 2001.

[AJ01] M. Abadi and J. Jurgens. Formal Eavesdropping and its Computational Interpretation. In *Proceedings of the 4th International Symposium on Theoretical Aspects of Computer Software (TACS)*, pp. 82–94, 2001.

[AR02] M. Abadi and P. Rogaway. Reconciling Two Views of Cryptography (The Computational Soundness of Formal Encryption). In *Journal of Cryptology*, vol. 15(2), pp. 103–127, 2002. (appeared also in *Proceedings of the 1st International Conference on Theoretical Computer Science (IFIP TCS)*, LNCS 1872, pp. 3–22, 2000.)

[AW05] M. Abadi, B. Warinschi. Password-Based Encryption Analyzed. In *Proceedings of the 32nd International Colloquium on Automata, Languages, and Programming (ICALP)*, LNCS 3580, pp. 664–676, 2005.

[Blu82] M. Blum. Coin Flipping by Phone. *24th IEEE Computer Society International Conference (COMPCON)*, pp. 133–137, 1982. (Also in *SIGACT News*, vol. 15(1), 1983.)

[BAN89] M. Burrows, M. Abadi, and R. Needham. A Logic of Authentication. In *Proceedings of the Royal Society of London A*, vol. 426, pp. 233–271, 1989.

[BCNP04] B. Barak, R. Canetti, J.B. Nielsen, and R. Pass. Universally Composable Protocols with Relaxed Set-Up Assumptions. In *Proceedings of the 45th IEEE Symposium on Foundations of Computer Science (FOCS)*, IEEE, pp. 186–195, 2004.

[BDJR97] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A Concrete Security Treatment of Symmetric Encryption: Analysis of the DES Modes of Operation. In *Proceedings of the 38th IEEE Symposium on Foundations of Computer Science (FOCS)*, IEEE, pp. 394–403, 1997.

[BF03] D. Boneh and M. Franklin. Identity Based Encryption from the Weil Pairing. *SIAM Journal on Computing*, vol. 32(3), pp. 586–615, 2003.

[BHS04] R. Bradshaw, J. Holt, and K. Seamons. Concealing Complex Policies with Hidden Credentials. In *Proceedings of the 11th ACM Conference on Computer and Communications Security (CCS)*, pp. 146–157, 2004.

[BKR00] M. Bellare, J. Kilian, and P. Rogaway. The Security of Cipher Block Chaining Message Authentication Code. In *Journal of Computer and System Sciences (JCSS)*, vol. 61(3), pp. 362–399, 2000.

[BM84] M. Blum and S. Micali. How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits. *SIAM Journal on Computing*, vol. 13(4), pp. 850–864, 1984.

[BM05] W. Bagga and R. Molva. Policy-Based Cryptography and Applications. In *Proceedings of the 9th International Conference on Financial Cryptography and Data Security*, LNCS 3570, pp. 72–87, 2005.

[BMC06] W. Bagga, R. Molva, and S. Crosta. Policy-Based Encryption Schemes from Bilinear Pairings. In P̃roceedings of the ACM Symposium on Information, Computer and Communications Security (ASIACCS), p. 368, 2006.

[BMR90] D. Beaver, S. Micali, and P. Rogaway. The Round Complexity of Secure Protocols. In *Proceedings of the 22nd ACM Symposium on Theory of Computing (STOC)*, ACM, pp. 503–513, 1990.

[BN00] M. Bellare and C. Namprempre. Authenticated Encryption: Relations Among Notions and Analysis of the Generic Composition Paradigm. In *Advances in Cryptology — ASIACRYPT 2000*, LNCS 1976, pp. 541–545, 2000.

[BP05] M. Backes and B. Pfitzmann. Relating Symbolic and Cryptographic Secrecy. In *Proceedings of the 26th IEEE Symposium on Security and Privacy (S&P)*, IEEE, pp. 171–182, 2005. Full version available at http://eprint.iacr.org/2004/300.

[BR93] M. Bellare and P. Rogaway. Entity Authentication and Key Distribution. In *Advances in Cryptology — CRYPTO 1993*, LNCS 773, pp. 232–249, 1993.

[BS00] P. Bonatti and P. Samarati. Regulating Service Access and Information Release on the Web. In *Proceedings of the 7th ACM Conference on Computer and Communications Security (CCS)*, pp. 134-143, 2000.

[Can01] R. Canetti. Universally Composable Security: A New Paradigm for Cryptographic Protocols. In *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science (FOCS)*, IEEE, pp. 136–145, 2001. Full version available at http://eprint.iacr.org/2000/067.

[Cha82] D. Chaum. Blind Signatures for Untraceable Payments. In *Advances in Cryptology — CRYPTO 1982*, LNCS, pp. 199–203, 1982.

[CCKM00] C. Cachin, J. Camenisch, J. Kilian, and J. Müller. One-Round Secure Computation and Secure Autonomous Mobile Agents. In *27th International Colloquium on Automata, Languages and Programming (ICALP)*, LNCS 1853, pp. 512–523, 2000.

[CDPW07] R. Canetti, Y. Dodis, R. Pass, and S. Walfish. Universally Composable Security with Global Setup. In *Theory of Cryptography Conference (TCC)*, 2007, to appear.

[CF01] R. Canetti and M. Fischlin. Universally Composable Commitments. In *Advances in Cryptology — CRYPTO 2001*, LNCS 2139, pp. 19–40, 2001.

[CH04] R. Canetti and J. Herzog. Universally Composable Symbolic Analysis of Mutual Authentication and Key-Exchange Protocols. In *Theory of Cryptography — TCC 2006*, LNCS 3876, pp. 380–403, 2006. Full version available at http://eprint.iacr.org/2004/334.

[CKL06] R. Canetti, E. Kushilevitz, and Y. Lindell. On the Limitations of Universally Composable Two-Party Computation Without Set-Up Assumptions. In *Journal of Cryptology*, vol. 19(2), pp. 135–167, 2006.

[CKPR01] R. Canetti, J. Kilian, E. Petrank, and A. Rosen. Concurrent Zero-Knowledge Requires $\widetilde{\Omega}(\log n)$ Rounds. In *Proceedings of the 33rd ACM Symposium on Theory of Computing (STOC)*, ACM, pp. 570–579, 2001.

[CLOS02] R. Canetti, Y. Lindell, R. Ostrovsky and A. Sahai. Universally Composable Two-Party and Multi-Party Secure Computation. In *Proceedings of the 34th ACM Symposium on Theory of Computing (STOC)*, ACM, pp. 494–503, 2002. Full version available at http://eprint.iacr.org/2002/140.

[CW79] J. Carter and M. Wegman. Universal Classes of Hash Functions. *Journal of Computer and System Sciences*, vol. 18(2), pp. 143–154, 1979.

[DDN00] D. Dolev, C. Dwork, and M. Naor. Non-Malleable Cryptography. *SIAM Journnal on Computing*, vol. 30(2), pp. 391–437, 2000.

[DDO$^+$01] A. de Santis, G. Di Crescenzo, R. Ostrovsky, G. Persiano and A. Sahai. Robust Non-Interactive Zero-Knowledge. In *Advances in Cryptology — CRYPTO 2001*, LNCS 2139, pp. 566–598, 2001.

[DH76] W. Diffie and M. Hellman. New Directions in Cryptography. In *IEEE Transactions on Information Theory*, vol. 22(6), pp. 644–654, 1976.

[DY83] D. Dolev and A.C. Yao. On the Security of Public Key Protocols. In *IEEE Transactions on Information Theory (IT)*, vol. 29(12), pp. 198–208, 1983.

[Fis02] M. Fischlin. On the Impossibility of Constructing Non-Interactive Statistically-Secret Protocols From any Trapdoor One-Way Function. *Cryptographers' Track — RSA 2002*, LNCS 2271, pp. 79–95, 2002.

[Fis06] M. Fischlin. Round-Optimal Composable Blind Signatures in the Common Reference String Model. In *Advances in Cryptology — CRYPTO 2006*, LNCS 4117, pp. 60–77, 2006.

[FAL06] K. B. Frikken, M. J. Atallah, and J. Li. Attribute-Based Access Control with Hidden Policies and Hidden Credentials. In *IEEE Transactions on Computers*, vol. 55(10), pp. 1259–1270, 2006.

[FGG$^+$06] M. Fitzi, J. Garay, S. Gollakota, C.P. Rangan, and K. Srinathan. Round-Optimal and Efficient Verifiable Secret Sharing. In *Theory of Cryptography — TCC 2006*, LNCS 3876, pp. 329–342, 2006.

[FL82] M.J. Fischer and N.A. Lynch. A Lower Bound for the Time to Assure Interactive Consistency. In *Information Processing Letters*, vol. 14(4), pp. 183–186, 1982.

[FLA06] K. B. Frikken, J. Li, and M. J. Atallah. Trust Negotiation with Hidden Credentials, Hidden Policies, and Policy Cycles. In *Proceedings of the 13th Annual Network and Distributed System Security Symposium (NDSS)*, pages 157–172, 2006.

[FM97] P. Feldman and S. Micali. An Optimal Probabilistic Protocol for Synchronous Byzantine Agreement. In *SIAM Journal of Computing*, vol. 26(4), pp. 873–933, 1997.

[Gol01] O. Goldreich. Foundations of Cryptography: Volume 1 – Basic Tools. Cambridge University Press, 2001.

[Gol04] O. Goldreich. Foundations of Cryptography: Volume 2 – Basic Applications. Cambridge University Press, 2004.

[GGKT05] R. Gennaro, Y. Gertner, J. Katz, and L. Trevisan. Bounds on the Efficiency of Generic Cryptographic Constructions. *SIAM Journal on Computing*, vol. 35(1), pp. 217–246, 2005.

[GGM84] O. Goldreich, S. Goldwasser, and S. Micali. On the Cryptographic Applications of Random Functions. In *Advances in Cryptology — CRYPTO 1984*, LNCS 196, pp. 276–288, 1984.

[GGM86] O. Goldreich, S. Goldwasser, and S. Micali. How to Construct Random Functions. In *Journal of the ACM*, ACM, vol. 33(4), pp. 792–807, 1986.

[GIKR01] R. Gennaro, Y. Ishai, E. Kushilevitz, and T. Rabin. The Round Complexity of Verifiable Secret Sharing and Secure Multicast. In *Proceedings of the 33rd ACM Symposium on Theory of Computing (STOC)*, ACM, pp. 580–589, 2001.

[GIKR02] R. Gennaro, Y. Ishai, E. Kushilevitz, and Tal Rabin. On 2-Round Secure Multiparty Computation. In *Advances in Cryptology — CRYPTO 2002*, LNCS 2442, pp. 178–193, 2002.

[GK96a] O. Goldreich and A. Kahan. How to Construct Constant-Round Zero-Knowledge Proof Systems for NP. In *Journal of Cryptology*, vol. 9(3), pp. 167–190, 1996.

[GK96b] O. Goldreich and H. Krawczyk. On the Composition of Zero-Knowledge Proof Systems. In *SIAM Journal of Computing*, vol. 25(1), pp. 169–192, 1996.

[GKM$^+$00] Y. Gertner, S. Kannan, T. Malkin, O. Reingold, and M. Viswanathan. The Relationship between Public-Key Encryption and Oblivious Transfer. *Proceedings of the 41st IEEE Symposium on Foundations of Computer Science (FOCS)*, IEEE, pp. 325–335, 2000.

[GL89] O. Goldreich and L. Levin. Hard-Core Predicates for any One-Way Function. *Proceedings of the 21st ACM Symposium on Theory of Computing (STOC)*, ACM, pp. 25–32, 1989.

[GM84] S. Goldwasser and S. Micali. Probabilistic Encryption. In *Journal of Computer and System Sciences (JCSS)*, vol. 28(2), pp. 270-299, 1984.

[GM95] J. W. Gray III and J. McLean. Using Temporal Logic to Specify and Verify Cryptographic Protocols (Progress Report). In *8th IEEE Computer Security Foundations Workshop*, pp. 108–116, 1995.

[GM98] J. Garay and Y. Moses. Fully Polynomial Byzantine Agreement for $n > 3t$ Processors in $t+1$ Rounds. In *SIAM Journal of Computing*, vol. 27(1), pp. 247–290, 1998.

[GMR88] S. Goldwasser, S. Micali, and R. Rivest. A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attack. In *SIAM Journal on Computing*, vol. 17, pp. 281–308, 1988.

[GMR89] S. Goldwasser, S. Micali and C. Rackoff. The Knowledge Complexity of Interactive Proof Systems. In *SIAM J. of Computing*, vol. 18(1), pp. 186–208, 1989.

[GMR01] Y. Gertner, T. Malkin, and O. Reingold. On the Impossibility of Basing Trapdoor Functions on Trapdoor Predicates. *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science (FOCS)*, IEEE, pp. 126–135, 2001.

[GMW87] O. Goldreich, S. Micali, and A. Wigderson. How to Play any Mental Game, or A Completeness Theorem for Protocols with Honest Majority. In *Proceedings of the 19th ACM Symposium on Theory of Computing (STOC)*, ACM, pp. 218–229, 1987.

[HBS03] J. E. Holt, R. W. Bradshaw, K. E. Seamons, and H. Orman. Hidden Credentials. In *Proceedings of the ACM Workshop on Privacy in the Electronic Society*, pp. 1–8, 2003.

[HILL99] J. Haståd, R. Impagliazzo, L. Levin, and M. Luby. A Pseudorandom Generator From any One-Way Function. *SIAM Journal on Computing*, vol. 28(4), pp. 1364–1396, 1999.

[HG03] O. Horvitz and V. Gligor. Weak Key Authenticity and the Computational Completeness of Formal Encryption. *Advances in Cryptology — CRYPTO 2003*, LNCS 2729, pp. 530–547, 2003.

[HK05] O. Horvitz and J. Katz. Bounds on the Efficiency of 'Black-Box' Commitment Schemes. In *32nd International Colloquium on Automata, Languages, and Programming (ICALP)*, LNCS 3580, pp. 128–139, 2005. Also in *Journal of Theoretical Computer Science (TCS)*, invited, to appear.

[HK07] O. Horvitz and J. Katz. Universally-Composable Two-Party Computation in Two Rounds. In *Advances in Cryptology — CRYPTO 2007*, LNCS 4622, 2007. To appear.

[HMQU05] D. Hofheinz, J. Müller-Quade, and D. Unruh. Universally Composable Zero-Knowledge Arguments and Commitments from Signature Cards. In *Proceedings of the 5th Central European Conference on Cryptology — MoraviaCrypt*, 2005.

[IK00] Y. Ishai and E. Kushilevitz. Randomizing Polynomials: A New Representation with Applications to Round-Efficient Secure Computation. In *Proceedings of the 41st IEEE Symposium on Foundations of Computer Science (FOCS)*, IEEE, pp. 294–304, 2000.

[IL89] R. Impagliazzo and S. Luby. One-Way Functions are Essential for Complexity-Based Cryptography. *Proceedings of the 30th IEEE Symposium on Foundations of Computer Science (FOCS)*, IEEE, pp. 230–235, 1989.

[IR89] R. Impagliazzo and S. Rudich. Limits on the Provable Consequences of One-Way Permutations. *Proceedings of the 21st ACM Symposium on Theory of Computing (STOC)*, ACM, pp. 44–61, 1989.

[JLO97] A. Juels, M. Luby, and R. Ostrovsky. Security of Blind Digital Signatures. In *Advances in Cryptology — CRYPTO 1997*, LNCS 1294, pp. 150–164, 1997.

[JS07] S. Jarecki and V. Shmatikov. Efficient Two-Party Secure Computation on Committed Inputs. In *Advances in Cryptology — EUROCRYPT 2007*, LNCS 4515, pp. 97–114, 2007.

[Katz07] J. Katz. Universally Composable Multi-Party Computation using Tamper-Proof Hardware. In *Advances in Cryptology — EUROCRYPT 2007*, LNCS 4515, pp. 115–128, 2007.

[Kem89] R. Kemmerer. Analyzing Encryption Protocols Using Formal Verifcation Techniques. In *IEEE Journal on Selected Areas in Communications*, vol. 7(4), pp. 448–457, 1989.

[KMM94] R. Kemmerer, C. Meadows, and J. Millen. Three Systems for Cryptographic Protocol Analysis. In *Journal of Cryptology*, vol. 7(2), pp. 79–130, 1994.

[KO04] J. Katz and R. Ostrovsky. Round-Optimal Secure Two-Party Computation. In *Advances in Cryptology — CRYPTO 2004*, LNCS 3152, pp. 335–354, 2004.

[KOS03] J. Katz, R. Ostrovsky, and A. Smith. Round Efficiency of Multi-Party Computation with a Dishonest Majority. In *Advances in Cryptology — EUROCRYPT 2003*, LNCS 2656, pp. 578–595, 2003.

[KST99] J.H. Kim, D.R. Simon, and P. Tetali. Limits on the Efficiency of One-Way Permutation-Based Hash Functions. *Proceedings of the 40th IEEE Symposium on Foundations of Computer Science (FOCS)*, IEEE, pp. 535–542, 1999.

[KY00] J. Katz and M. Yung. Unforgeable Encryption and Chosen Ciphertext Secure Modes of Operation. In *Proceedings of the 7th International Workshop on Fast Software Encryption (FSE)*, LNCS 1978, pp. 284–299, 2000.

[Lin01] Y. Lindell. Parallel Coin-Tossing and Constant-Round Secure Two-Party Computation. In *Journal of Crypto*, vol. 16(3), pp. 143–184, 2003.

[Llo87] J. W. Lloyd. *Foundations of Logic Programming.* Second Edition, Springer-Verlag, 1987.

[Low96] G. Lowe. Breaking and Fixing the Needham-Schroeder Public-Key Protocol Using FDR. In *Tools and Algorithms for the Construction and Analysis of Systems*, LNCS 1055, pp. 147–166, 1996.

[LDB03] N. Li, W. Du, and D. Boneh. Oblivious Signature-Based Envelope. In *Proceedings of the 22nd ACM Symposium on Principles of Distributed Computing (PODC)*, pp. 182–189, 2003.

[LL05a] J. Li and N. Li. OACerts: Oblivious Attribute Certificates. In *Proceedings of the 3rd Conference on Applied Cryptography and Network Security (ACNS)*, LNCS 3531, pp. 301–317, 2005.

[LL05b] J. Li and N. Li. Policy-Hiding Access Control in Open Environments. In *Proceedings of the 24nd ACM Symposium on Principles of Distributed Computing (PODC)*, pp. 29–38, 2005.

[LL06] J. Li and N. Li. A Construction for General and Efficient Oblivious Commitment Based Envelope Protocols. In *Proceedings of 8th International Conference on Information and Communications Security (ICICS)*, pp. 122–138, 2006.

[LN84] J. L. Lassez, V. L. Nguyen, E. A. Sonenberg. Fixpoint Theorems and Semantics: a Folk Tale In *Information Processing Letters*, vol. 14(3), pp. 112–116, 1982,.

[LP04] Y. Lindell and B. Pinkas. A Proof of Yao's Protocol for Secure Two-Party Computation. In *Journal of Cryptology*, to appear. Full version available at http://eprint.iacr.org/2004/175.

[Mea91] C. Meadows. A System for the Specifcation and Analysis of Key Management Protocols. In *Proceedings of the 12th IEEE Symposium on Research in Security and Privacy (S&P)*, pp. 182–195, 1991.

[MCF87] J. K. Millen, S. C. Clark, and S. B. Freedman. The Interrogator: Protocol Security Analysis. In *IEEE Transactions on Software Engineering (TSE)*, vol. 13(2), pp. 274–288, 1987.

[MMS97] J. C. Mitchell, M. Mitchell, and U. Stern. Automated Analysis of Cryptographic Protocols Using Murϕ. In *Proceedings of the 18th IEEE Symposium on Security and Privacy (S&P)*, pp. 141–151, 1997.

[MP05] D. Micciancio and S. Panjwani. Adaptive Security of Symbolic Encryption. In *Theory of Cryptography — TCC 2005*, LNCS 3378, pp. 169–187, 2005.

[MW04a] D. Micciancio and B. Warinschi. Completeness Theorems for the Abadi-Rogaway Logic of Encrypted Expressions. In *Journal of Computer Security*, vol. 12(1), pp. 99–129, 2004.

[MW04b] D. Micciancio and B. Warinschi. Soundness of Formal Encryption in the Presence of Active Adversaries In *Theory of Cryptography — TCC 2004*, LNCS 2951, pp. 133–151, 2004.

[Nao91] M. Naor. Bit Commitment Using Pseudorandomness. *Journal of Cryptology*, vol. 4(2), pp. 151–158, 1991.

[NP01] M. Naor and B. Pinkas. Efficient Oblivious Transfer Protocols. In *Proceedings of the 12th Symposium on Discrete Algorithms (SODA)*, pp. 448–457, 2001.

[NT05] S. Nasserian and G. Tsudik. Revisiting Oblivious Signature-Based Envelopes. Available at http://eprint.iacr.org/2005/283.

[NY89] M. Naor and M. Yung. Universal One-Way Hash Functions and their Cryptographic Applications. *Proceedings of the 21st ACM Symposium on Theory of Computing (STOC)*, ACM, pp. 33–43, 1989.

[Pau98] L. Paulson. The Inductive Approach to Verifying Cryptographic Protocols. In *Journal of Computer Security*, vol. 6(1–2), pp. 85–128, 1998.

[PRS02] M. Prabhakaran, A. Rosen, and A. Sahai. Concurrent Zero Knowledge with Logarithmic Round-Complexity. In *Proceedings of the 43rd IEEE Symposium on Foundations of Computer Science (FOCS)*, IEEE, pp. 366–375, 2002.

[PSL80] M. Pease, R. Shostak, and L. Lamport. Reaching Agreement in the Presence of Faults. In *Journal of the ACM*, vol. 27(2), pp. 228–234, 1980.

[PW00] B. Pfitzmann and M. Waidner. Composition and Integrity Preservation of Secure Reactive Systems. In *Proceedings of the 7th ACM Conference on Computer and Communications Security (CCS)*, pp. 245–254, 2000.

[Rom90] J. Rompel. One-Way Functions are Necessary and Sufficient for Secure Signatures. *Proceedings of the 22nd ACM Symposium on Theory of Computing (STOC)*, ACM, pp. 387–394, 1990.

[Rud88] S. Rudich. Limits on the Provable Consequences of One-Way Functions. Ph.D. thesis, University of California at Berkeley, 1988.

[Rud91] S. Rudich. The Use of Interaction in Public Cryptosystems. *Advances in Cryptology — CRYPTO 1992*, LNCS 576, pp. 242–251, 1992.

[RTV04] O. Reingold, L. Trevisan, and S. Vadhan. Notions of Reducibility Between Cryptographic Primitives. *Theory of Cryptography — TCC 2004*, LNCS 2951, pp. 1–20, 2004.

[Sim98] D.R. Simon. Finding Collisions on a One-Way Street: Can Secure Hash Functions be Based on General Assumptions? *Advances in Cryptology — EUROCRYPT 1403*, LNCS 334, pp. 345–98, 1403.

[SWY01] K. E. Seamons, M. Winslett, and T. Yu. Limiting the Disclosure of Access Control Policies During Automated Trust Negotiation. In *Proceed-*

ings of the 8th Annual Network and Distributed System Security Symposium (NDSS), 2001.

[SYY99] T. Sander, A. Young, and M. Yung. Non-Interactive CryptoComputing For NC^1. In *Proceedings of the 40th IEEE Symposium on Foundations of Computer Science (FOCS)*, IEEE, pp. 554–567, 1999.

[Tar55] A. Tarski. A Lattice-Theoretical Fixpoint Theorem and its Applications. In *Pacific Journal of Mathematics*, vol. 5, pp.285–309, 1955.

[Tau05] Y. Tauman Kalai. Smooth Projective Hashing and Two-Message Oblivious Transfer. In *Advances in Cryptology — EUROCRYPT 2005*, LNCS 3494, pp. 78–95, 2005.

[THG99] F. Javier Thayer, J. C. Herzog, and J. D. Guttman. Strand Spaces: Proving Security Protocols Correct. In *Journal of Computer Security*, vol. 7(1), 1999.

[WL04] W. H. Winsborough and N. Li. Safety in Automated Trust Negotiation. In *Proceedings of the 25th IEEE Symposium on Security and Privacy (S&P)*, IEEE, pp. 147–160, 2004.

[Yao82] A.C.-C. Yao. Theory and Application of Trapdoor Functions. *Proceedings of the 23rd IEEE Symposium on Foundations of Computer Science (FOCS)*, IEEE, pp. 80–91, 1982.

[Yao86] A.C.-C. Yao. How to Generate and Exchange secrets. In *Proceedings of the 27th IEEE Symposium on Foundations of Computer Science (FOCS)*, IEEE, pp. 162–167, 1986.

[YW03] T. Yu and M. Winslett. Unified Scheme for Resource Protection in Automated Trust Negotiation. In *Proceedings of the 24th IEEE Symposium on Security and Privacy (S&P)*, IEEE, pp. 110–122, 2003.

[YWS01] T. Yu, M. Winslett, and K. E. Seamons. Interoperable Strategies in Automated Trust Negotiation. In *Proceedings of the 8th ACM Conference on Computer and Communications Security (CCS)*, pp. 146-155, 2001.